



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Gefördert durch:



Bundesministerium
für Wirtschaft
und Energie

aufgrund eines Beschlusses
des Deutschen Bundestages

KI Embedded

KI-Grundlagenentwicklung mit Leitanwendungen Virtuelle Sensorik und Brennstoffzellenregelung

Teilvorhaben:

Entwicklung effizienter, KI-gestützter, modellprädiktiver Regelung

Schlussbericht

Zuwendungsempfänger	Technische Universität Darmstadt
Förderkennzeichen	19I21043G
Projektlaufzeit	01.09.2021 – 31.12.2024

Das diesem Bericht zugrunde liegende Vorhaben wurde mit Mitteln des Bundesministeriums für Wirtschaft und Energie gefördert.

Die Verantwortung für den Inhalt dieser Veröffentlichung liegt bei den Autoren.

Autoren

Prof. Dr.-Ing. Rolf Findeisen

Hendrik Alsmeier, M. Sc.

Dr.-Ing. Eric Lenz

Alexander Rose, M. Sc.

Inhaltsverzeichnis

1	Kurze Darstellung	3
1.1	Aufgabenstellung	3
1.2	Voraussetzungen, unter denen das Vorhaben durchgeführt wurde	3
1.3	Planung und Ablauf des Vorhabens	4
1.3.1	Projektaufbau	4
1.3.2	Veränderungen gegenüber der ursprünglichen Planung	4
1.4	Wissenschaftlicher und technischer Stand	5
1.5	Zusammenarbeit mit anderen Stellen	6
2	Eingehende Darstellung	7
2.1	Verwendung der Zuwendung und erzielte Ergebnisse	7
2.1.1	AP1.TUD	7
2.1.2	AP5.TUD	29
2.2	Wichtigste Positionen des zahlenmäßigen Nachweises	42
2.3	Notwendigkeit und Angemessenheit der geleisteten Arbeit	42
2.4	Voraussichtlicher Nutzen, insbesondere der Verwertbarkeit des Ergebnisses im Sinne des fortgeschriebenen Verwertungsplans	42
2.5	Während der Durchführung des Vorhabens dem ZE bekannt gewordener Fortschritt auf dem Gebiet des Vorhabens bei anderen Stellen	42
2.6	Veröffentlichungen	44
	Literatur	45

1 Kurze Darstellung

Dieser Schlussbericht beschreibt die Arbeiten des Fachgebietes Regelung und Cyberphysische Systeme (Control and Cyber-Physical Systems, CCPS) der Technischen Universität Darmstadt (TUDa) in dem Verbundprojekt KI-Embedded (KI-Grundlagenentwicklung mit Leitanwendungen Virtuelle Sensorik und Brennstoffzellenregelung) im Zeitraum von September 2021 bis Dezember 2024.

1.1 Aufgabenstellung

Modellbasierte Regler erlauben es, komplexe Prozesse optimal führen zu können. Jedoch sind sie, trotz erheblicher Fortschritte der letzten Jahre, bisher nur bedingt auf eingebetteten Systemen einsetzbar. Im Verbundprojekt widmet sich das IAT mit dem Fachgebiet Regelungstechnik und Cyberphysische Systeme insbesondere der Entwicklung effizienter KI-basierter Ansätze für die optimierungsbasierte prädiktive Regelung auf eingebetteten Systemen, sowie der exemplarischen Umsetzung einer Toolkette, die es erlaubt die entwickelten Verfahren schnell und effizient, unter Ausnutzung der Hardwarebeschleuniger auf eingebetteten Steuergeräten anzuwenden.

Die zu entwickelnden und implementierenden Ansätze und Methoden sollen einerseits Garantien für die Leistungsfähigkeit und Zuverlässigkeit geben können und müssen andererseits direkt auf Steuergeräten einsetzbar sein. Diese Ansätze und Werkzeuge werden mit dem Partner exemplarisch anhand des Use-Cases Brennstoffzellenstack evaluiert. Neben der Entwicklung geeigneter, echtzeitfähiger Ansätze, entsteht eine exemplarische Toolkette zur Umsetzung auf eingebetteten Systemen, sowie eine KI-unterstützte Open-Source Toolbox zur Simulation, Analyse und Regelung von Brennstoffzellenstacks.

1.2 Voraussetzungen, unter denen das Vorhaben durchgeführt wurde

Der sichere und ökonomische Betrieb einer Brennstoffzelle, welcher Voraussetzung für einen wirtschaftlichen Einsatz ist, stellt jedoch Herausforderungen bereit. Hier sind zum einen während des Betriebs verschiedene Grenzen an Eingangsgrößen und innere Systemgrößen der Brennstoffzelle einzuhalten, damit diese keinen Schaden nimmt. Hierfür bietet sich der Ansatz einer modellprädiktiven Regelung (MPC, model predictive control) an. Jedoch ist das Modell einer Brennstoffzelle relativ komplex, und die zur Verfügung stehende Rechenleistung auf Steuergeräten dagegen gering, sodass der direkte Einsatz problematisch ist.

Ein Ansatz zur Reduktion der Komplexität besteht im Einsatz von Methoden aus dem Gebiet der Künstlichen Intelligenz (KI).

Die Fortschritte im Bereich der KI der vorangegangenen Jahre zeigt das große Potential, das in diesem Bereich steckt.¹ Jedoch sind die dabei zum Einsatz kommenden Technologien für den hier betrachteten Anwendungsfall nicht direkt einsetzbar, da auf den Steuergeräten (eingebettete Systeme), die hier

¹Dies wird auch mit den während der Projektlaufzeit allgemein zugänglich gewordenen, einfach verfügbaren KI-Anwendungen, wie z. B. LLMs (Large Language Models) von denen hier nur ChatGPT von OpenAI als ein kommerzielles Beispiel genannt werden soll, nochmals deutlich und bestätigt diese Annahme.

betrachtet werden, nur eine, im Vergleich zu GPU-Servern oder ähnlichen Systemen, verschwindend kleine Rechenleistung zur Verfügung steht. Um das Potenzial der KI auch für diese Bereiche zu nutzen, sind daher weitere Forschungsanstrengungen notwendig, die durch den Beitrag des CCPS innerhalb dieses Gesamtprojektes adressiert werden.

Das Fachgebiet CCPS der TUDa mit dem Leiter Prof. Rolf Findeisen² verfügt über eine ausgewiesene, international anerkannte Expertise auf dem Gebiet der modellprädiktiven Regelung, insbesondere auch unter Berücksichtigung von Ansätzen aus dem Bereich der KI.

1.3 Planung und Ablauf des Vorhabens

1.3.1 Projektaufbau

Abbildung 1.1 zeigt eine (dem Gesamtbericht entnommene) Übersicht über die Themen, die im Rahmen des Gesamtprojektes bearbeitet wurden.

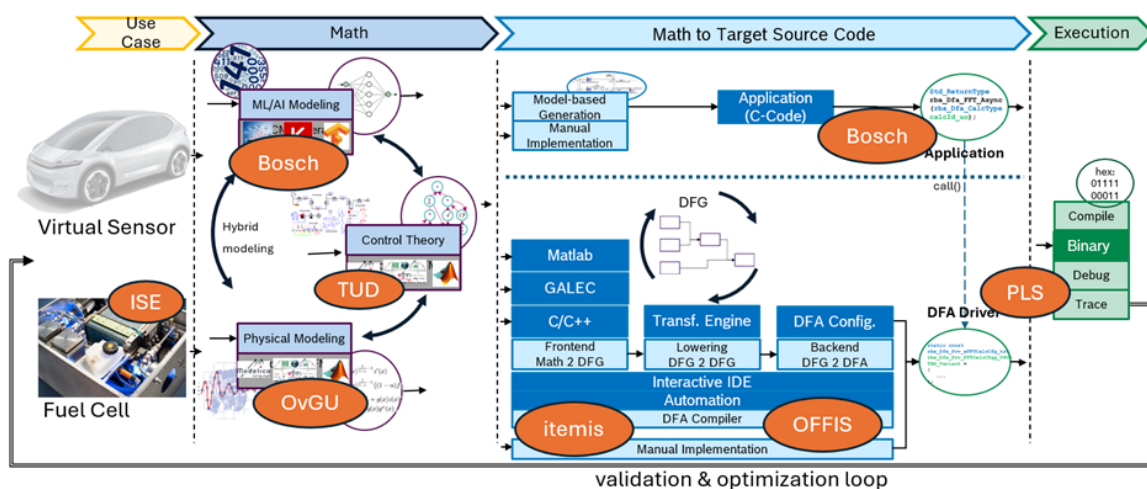


Abbildung 1.1: Themen Gesamtprojekt [dem Gesamtbericht entnommen]

Abbildung 1.2 zeigt die sich daraus ergebende Einteilung in sechs Arbeitspakete. Die TUDa war hiervon in zwei Arbeitspaketen involviert.

Die wesentliche methodische Arbeit wurde im Rahmen des Arbeitspakets 1 durchgeführt. Hierbei wurden alle Arbeiten im Rahmen des Unterarbeitspaketes (UAP) 1.2 (Konzepte zur effizienten, KI-gestützten, modellprädiktiven Regelung) durchgeführt.

Im Arbeitspaket 5 erfolgt die exemplarische Umsetzung am Beispiel der Regelung einer Brennstoffzelle. Hier war die TUDa in den Teilarbeitspaketen UAP5.1 (Modellierung) und UAP5.4 (Validierung) involviert und hat darüber hinaus alle Arbeiten im UAP5.2 (Reglerentwurf) durchgeführt.

1.3.2 Veränderungen gegenüber der ursprünglichen Planung

Während des Projektes haben sich einige Verzögerungen ergeben. So hat insbesondere die Einstellung neuer Projektmitarbeiter zu Beginn aufgrund der Covid-Pandemie und des Wechsels von Prof. Findeisen an die TUDa einen längeren Zeitraum in Anspruch genommen als geplant war. Während der

²Bei Antragsstellung noch Leiter des Lehrstuhls für Systemtheorie und Regelungstechnik der Otto-von-Guericke-Universität in Magdeburg.

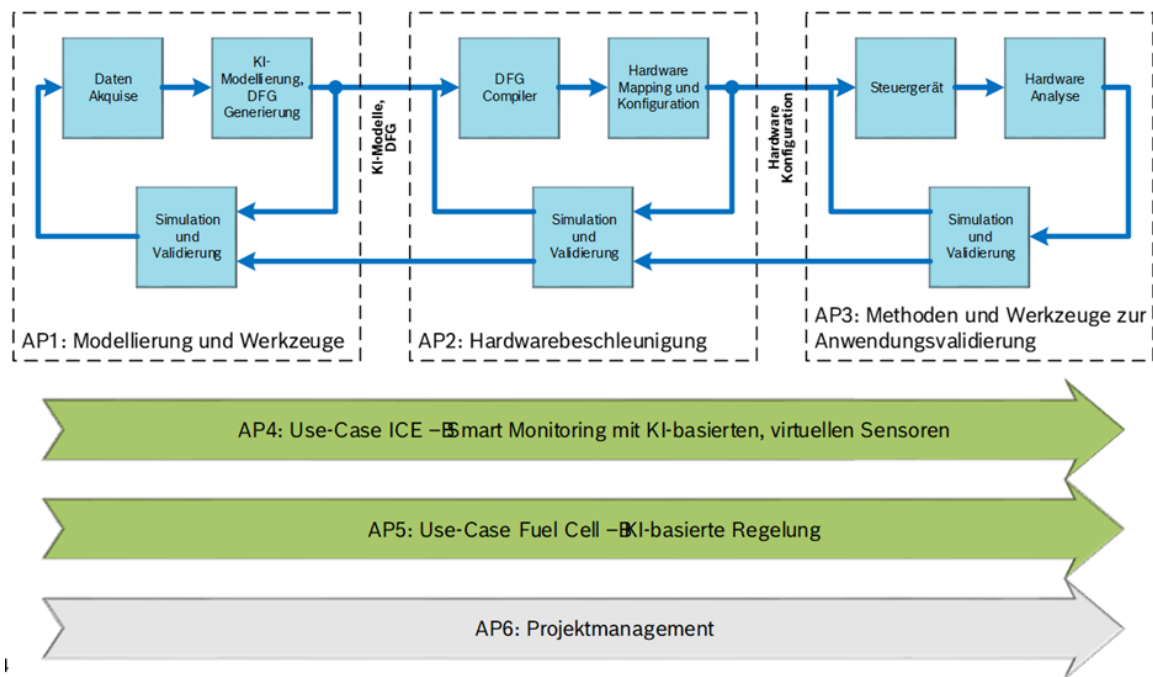


Abbildung 1.2: Arbeitspakete Gesamtprojekt [dem Gesamtbericht entnommen]

Projektlaufzeit hat es darüber hinaus eine unerwartete Fluktuation an Mitarbeitern gegeben, die ebenfalls nur mit Verzögerung ausgeglichen werden konnte. Zuletzt sind bei der Lieferung des Brennstoffzellenstacks erhebliche Verzögerungen aufgetreten.

Da auch die Projektpartner mit ähnlichen Problemen zu kämpfen hatten, wurde sich dazu entschieden, beim Projektträger einen Antrag auf eine Verlängerung der Projektlaufzeit um sechs Monate zu stellen. Diesem wurde gefolgt, so dass sich das Projektende von ursprünglich Juni 2024 auf Dezember 2024 verschoben hat.

Inhaltlich hat es keine Veränderungen bezüglich der ursprünglichen Planung gegeben.

1.4 Wissenschaftlicher und technischer Stand

Der Nutzen von KI-Modellen wie neuronalen Netzen und Gauß'schen Prozessen im Kontext der modellprädiktiven Regelung (Model Predictive Control, MPC) ist ein Ansatz, der in der wissenschaftlichen Gemeinschaft zunehmend an Bedeutung gewinnt. Entsprechend können wir in diesem Projekt bei der Entwicklung unserer neuen Ansätze auf eine Vielzahl relevanter Vorarbeiten aufbauen.

Ein wesentlicher Forschungszweig beschäftigt sich mit der Einbettung datengetriebener Modelle in den MPC Kontext. Dabei kommen insbesondere Gauß'sche Prozesse (GPs) und neuronale Netze (NNs) zum Einsatz. Gauß'sche Prozesse bieten neben ihrer Fähigkeit zur Modellierung nichtlinearer Dynamiken den Vorteil, dass sie eine explizite Quantifizierung der Modellunsicherheit ermöglichen. Durch diese Eigenschaft sind GPs besonders für Anwendungen, bei denen Sicherheitsaspekte eine zentrale Rolle spielen, geeignet. In der Forschung wurden hierfür umfassende Methoden entwickelt, um GPs sowohl zur Zustandsvorhersage als auch zur sicheren Formulierung probabilistischer Nebenbedingungen zu nutzen. Zusätzlich zeigen praktische Anwendungen, dass sich GP-Modelle, auch unter Echtzeitbedingungen, effektiv mit MPC kombinieren lassen etwa zur robusten Pfadverfolgung bei mobilen Systemen.

Neuronale Netze hingegen eignen sich insbesondere für Systeme mit komplexer und hochdimensionaler Dynamik, bei denen klassische Modellierungsansätze an ihre Grenzen stoßen. Dabei wird typischerweise ein neuronales Netz auf Basis von Systemdaten trainiert, um als dynamischer Prädiktor innerhalb des MPC zu fungieren. Moderne Ansätze verfolgen hierbei das Ziel, die Netzstruktur und das Training so auszulegen, dass die Vorhersagequalität über mehrere Zeitschritte hinweg erhalten bleibt und die Rechenzeit innerhalb der MPC-Schleife begrenzt wird. In aktuellen Arbeiten werden hierfür auch spezielle Architekturen wie Operatornetze oder physikinformierte Modelle eingesetzt. Diese ermöglichen es, die Vorteile datengetriebener Lernverfahren mit stabilitätsfördernden Eigenschaften klassischer Modellstrukturen zu verbinden.

Ein übergreifender Trend besteht darin, Unsicherheiten aus den datengetriebenen Modellen gezielt in das Optimierungsproblem des MPC Algorithmus zu integrieren. So lassen sich robuste oder probabilistische Varianten der MPC formulieren, die trotz unvollständigen Wissens über die Systemdynamik eine sichere Regelung ermöglichen. Dabei kommen verschiedene Strategien zum Einsatz, etwa die explizite Modellierung der Unsicherheit, konservative Eingangs- oder Zustandsbeschränkungen sowie adaptive Mechanismen zur Echtzeitanpassung des Modells. Der Fokus liegt hierbei auf der Verbindung von Lernfähigkeit und Verlässlichkeit, mit dem Ziel, datengetriebene MPC-Systeme auch in sicherheitskritischen oder ressourcenbeschränkten Anwendungen einsetzen zu können.

Diese Entwicklungen verdeutlichen, dass datengetriebene Modelle nicht nur eine leistungsfähige Alternative zu klassischen physikbasierten Systemmodellen darstellen, sondern auch gezielt in den regelungstechnischen Kontext integriert werden können. Voraussetzung hierfür ist, dass die damit einhergehenden Unsicherheiten systematisch adressiert werden.

Die konkreten Vorarbeiten sind bei der Vorstellung der Ergebnisse im zweiten Teil dieses Schlussberichtes jeweils genannt.

1.5 Zusammenarbeit mit anderen Stellen

Es haben halbjährlich Präsenztreffen über jeweils zweieinhalb Tage aller Projektpartner stattgefunden. Daneben erfolgte monatlich eine Online-Absprache der Projektleitenden.

Innerhalb der in AP5 (Anwendungsbeispiel Brennstoffzelle) involvierten Partner gab es eine zweiwöchentlichen Regeltermin, der ebenfalls online durchgeführt wurde. Da es bei letztgenannter Gruppe eine große Überschneidung mit den Gruppen gab, die in AP1 die systematischen Themen bearbeitet hatten, wurden dieses Treffen auch zum Austausch hierüber genutzt.

In der ersten Projekthälfte hat darüber hinaus eine sehr intensive Zusammenarbeit mit der OvGU stattgefunden, in der die Modellierung der Brennstoffzelle diskutiert wurde

2 Eingehende Darstellung

2.1 Verwendung der Zuwendung und erzielte Ergebnisse

Dieser Abschnitt beschreibt die Arbeiten, die mit den Personal- und Sachmitteln der Zuwendung durchgeführt wurden, sowie die erreichten Ergebnisse. Um Weitläufigkeit zu vermeiden, werden zu den Arbeiten und Ergebnissen, die schon von uns veröffentlicht wurden (Abschnitt 2.6), hier nur die wesentlichen Ideen erläutert und auf die entsprechenden Publikationen verwiesen.

2.1.1 AP1.TUD

UAP1.2.TUD.1 (Rechnerische und Betriebliche Anforderungen) Zu Beginn des Projektes arbeiteten wir eng mit Projektpartnern zusammen, um die rechnerischen und betrieblichen Anforderungen für die Integration von Methoden des maschinellen Lernens (ML) mit der modellprädiktiven Regelung in eingebetteten Systemen zu definieren. Als Ziel wurde definiert, skalierbare und effiziente MPC-Strategien zu entwickeln, welche auch auf einem Brennstoffzellen-Stack angewendet werden können. Zu Beginn musste das Ziel der Regelung klar definiert werden, wobei zwischen Sicherheit und Effizienz abgewogen werden musste. Hierbei ist zu bedenken, dass die Brennstoffzelle zu Schaden kommen kann, falls gewisse Grenzwerte über oder unterschritten werden. Dies steht jedoch bei gewissen Bedingungen der effizientesten Operation entgegen. Aus diesen Erwägungen wurde die Gewährleistung eines stabilen und sicheren Betriebs von Brennstoffzellen unter Einhaltung kritischer Randbedingungen wie Temperatur-, Spannungs- und Druckgrenzen als vorrangig definiert. Dies erachteten wir als sinnvoll, da bei dem Betrieb einer Brennstoffzelle über mehrere Jahre, z. B. in einem Fahrzeug, der Verschleiß minimiert werden sollte. Es wurde weiterhin entschieden, dass Effizienz speziell während transients Bedingungen, z. B. bei Sollwertänderungen, ein nachrangiges Kriterium darstellt. Effizienz nicht außer Acht zu lassen, ist im Hinblick auf Brennstoffreduktion sinnhaft, um die Ressourcen voll auszunutzen. Dies stellte einen idealen Anwendungsfall für MPC dar, welche zwischen verschiedenen Prioritäten abwägen und währenddessen immer noch optimale Stellgrößen vorgeben und Beschränkungen beachten kann. Als Basis für diese Entscheidung wurde eine umfassende Literaturrecherche zu bekannten Regelungsansätzen für Brennstoffzellen vorgenommen [19], [24]–[26], [35], [37], [44], [53], [57].

Auf das Anwendungsbeispiel der Regelung eines Brennstoffzellenstacks wird in Abschnitt 2.1.2 zu AP5 genauer eingegangen.

UAP1.2.TUD.2 (Rechen- und Speicherbedarf MPC und ML) Zum besseren Verständnis der Betrachtungen zu diesem Unterarbeitspaket folgt eine kurze Einführung zu MPC. Ein MPC-Algorithmus ist ein optimierungsbasierter Regelungsalgorithmus, welcher ein Optimalsteuerungsproblem (OCP, optimal control problem), im Allgemeinen ein nichtlineares Optimierungsproblem, während der Laufzeit lösen muss. Unter einem OCP verstehen wir speziell die Bestimmung einer Stellgrößenfolge $u_{0:N-1}$, die für das nichtlineare Zustandsmodell

$$x_{k+1} = f(x_k, u_k), \quad k = 0, \dots, N-1, \quad (2.1)$$

den Zielfunktionswert

$$\min_{u_{0:N-1}} J(X, U) \quad (2.2)$$

unter Einhaltung von Zustands- und Eingangsbeschränkungen

$$x_k \in \mathcal{X}, u_k \in \mathcal{U} \quad (2.3)$$

minimiert. Hier stellen $x_k \in \mathbb{R}^{n_x}$ und $u_k \in \mathbb{R}^{n_u}$ jeweils die Zustands- und Stellgrößen dar, wobei $k \in \mathbb{N}$ der diskrete Zeitindex ist. Der Transitionoperator $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_x}$ gibt vor, welcher Zustand, gegeben einem vorherigen Zustandswert und Stellgrößenwert, zum nächsten Abtastzeitpunkt erreicht wird. Die Kostenfunktion $J : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow [0, \infty)$ misst den Abstand der erreichten Werte zu den optimalen, gewünschten Werten. Die Mengen $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ und $\mathcal{U} \subseteq \mathbb{R}^{n_u}$ stellen die Bereiche der erlaubten Zustands- und Stellgrößenwerte dar. MPC löst ein solches Problem iterativ im Regelkreis, indem bei jedem Abtastschritt ein endlicher Horizont mittels f in die Zukunft prädiziert, J unter den Beschränkungen minimiert und nur der erste Stellgrößenwert appliziert wird. So lassen sich explizite Kompromisse z. B. zwischen Dynamik, Wirkungsgrad und Lebensdauer direkt in der Kostenfunktion abbilden [46]. Das gesamte OCP kann wie folgt geschrieben werden:

$$\begin{aligned} J^* = \min_U \quad & J(U, X), \\ \text{s.t.} \quad & \\ & x_{k+1} = f(x_k, u_k) \quad \forall k \in \{0, 1, \dots, N-1\}, \\ & g(x_k, u_k) \geq 0 \quad \forall k \in \{0, 1, \dots, N-1\}, \\ & x_k \in \mathcal{X} \quad \forall k \in \{0, 1, \dots, N-1\}, \\ & u_k \in \mathcal{U} \quad \forall k \in \{0, 1, \dots, N-1\}, \\ & x_N \in \mathcal{X}_T, \\ & x_0 = x_{\text{init}}. \end{aligned} \quad (2.4)$$

Hier werden mit $X = (x_0, x_1, \dots, x_N)$ und $U = (u_0, u_1, \dots, u_{N-1})$ jeweils die Zustände und Stellgrößen über den Regelunghorizont mit Länge $N \in \mathbb{N}$ beschrieben. Des Weiteren werden zusätzliche Beschränkungen mittels $g : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_g}$ eingeschlossen, $\mathcal{X}_T \subseteq \mathcal{X}$ gibt den gewünschten Bereich am Ende der Prädiktion an und x_{init} ist der gemessene Zustand des System zum aktuellen Abtastzeitpunkt und somit zu Beginn der Prädiktion in der Lösung des Optimierungsproblems.

Basierend auf dieser allgemeingültigen Definition eines MPC-Reglers können wir nun detailliertere Betrachtungen über dessen algorithmische Komplexität anstellen. Hierfür sind in erster Linie die mathematischen Eigenschaften der Kostenfunktion sowie der Gleichungs- und Ungleichungsbeschränkungen ausschlaggebend, da sie die Struktur des resultierenden Optimierungsproblems und damit den Rechenaufwand maßgeblich bestimmen. Zusätzlich vergrößern die Dimensionen des Zustands- (n_x) und Eingangsvektors (n_u) den Suchraum und wirken sich damit unmittelbar auf die Laufzeit aus.

Im quadratisch-linearen Fall sind die Kosten meist wie folgt definiert:

$$J = \sum_{k=0}^{N-1} (x_k^\top Q x_k + u_k^\top R u_k) + x_N^\top P x_N. \quad (2.5)$$

Sind System- bzw. Beschränkungsfunktionen f und g linear, so fällt das Optimalsteuerungsproblem (2.4) in die Klasse der Quadratischen Programme (QP). Eine Lösung eines QPs in Polynomialzeit ist nur garantiert, wenn J konvex ist. Andernfalls wird das Problem als NP-vollständig eingestuft [11].

Ist das Problem konvex, erreichen moderne Echtzeit-Implementierungen unter Nutzung der blockband-strukturierten KKT-Matrix für einen Prädiktionshorizont N eine sparse Riccati-Rekursion mit der Komplexität

$$\mathcal{O}(N(n_x + n_u)^3)$$

pro Newton-Iteration. Unter weiterer Ausnutzung der Separierbarkeit und eines Warmstarts lässt sich dieser Aufwand bei festem N auf

$$\mathcal{O}((n_x + n_u)^3)$$

reduzieren, so dass die Kosten mit der Länge des Horizonts lediglich linear skalieren [22]. In hochgradig eingebetteten Anwendungen kommen außerdem maßgeschneiderte Code-Generatoren (z. B. CVX-GEN oder FORCES-Pro) zum Einsatz, die eine statische Speicherzuweisung vornehmen und so Echtzeitfähigkeit auf Mikrocontrollern ermöglichen.

Weichen f , g oder J von der linearen bzw. quadratischen Annahme ab, entsteht ein Nichtlineares Programm (NLP). Dessen globale Lösung ist generell NP-schwer; dennoch führen Sequenzielle Quadratische Programmierung (SQP) oder Interior-Point-Methoden in der Praxis häufig zu akzeptablen lokalen Optima. Die typische Komplexität pro Iteration beträgt

$$\mathcal{O}(N(n_x + n_u)^3)$$

für eine stark ausgedünnte KKT-Matrix [55]. Dies kann jedoch nie garantiert werden und ist ein rein empirisches Kriterium. Wird zusätzlich die sogenannte Real-Time Iteration (RTI) eingesetzt, [20], genügt oft eine einzige Newton-Iteration je Abtastschritt, sofern sich die Problemparameter nur geringfügig ändern. Hierzu mehr unter UAP1.2.TUD.3.

Kommen diskrete Entscheidungen hinzu, wie etwa binäre Ventilschaltungen oder Gangwahl bei Hybridfahrzeugen, spricht man von Mixed-Integer MPC (MIQP bzw. MINLP). Derartige Probleme erfordern kombinatorische Suche (Branch-and-Bound oder Branch-and-Cut) und haben im Worst-Case eine exponentielle Laufzeit in der Anzahl der ganzzahligen Variablen. In der Praxis bewährt sich eine hierarchische Aufteilung: Zuerst wird auf einer groben Zeitskala eine diskrete Sequenz bestimmt, anschließend ein kontinuierliches QP gelöst.

Die praktischen Maßnahmen zur Komplexitätsreduktion werden später unter UAP1.2.TUD.3 näher beleuchtet werden. Jedoch verdeutlichen diese Betrachtungen, dass eine MPC im besten Falle eine Komplexität von $\mathcal{O}((n_x + n_u)^3)$ aufweist.

Zur Betrachtung der Komplexität von ML-Methoden haben wir GPs und NNs evaluiert. Hierbei wurde durch eine weitere Literaturrecherche belegt, dass sich beide Methoden mit MPC kombinieren lassen, wobei zwei Optionen bestehen: Das Lernen des Modells [23], [33], [47] und das Lernen des gesamten Reglers bzw. des Regelgesetzes [8], [17], [32]. Diese beiden Möglichkeiten bieten folgende Vorteile:

- **Modell:** Datengetriebene Teilmodelle (etwa neuronale Netze oder Gaussian Processes) approximieren schwer zugängliche Subsysteme, wie Membranbefeuchtung oder Kompressorkennfelder und erhöhen so die Prädiktionsgüte der Modelle. Andersherum können ML-Algorithmen eine Reduzierte Abbildung komplexer Modelle lernen, um diese für die MPC nutzbar machen zu können.
- **Regelgesetz:** Offline trainierte ML-Modelle approximieren die Optimallösung direkt (explicit / learning-based MPC). Dadurch reduziert sich die Online-Rechenlast von $\mathcal{O}(10^3 - 10^4)$ Gleitkommaoperationen auf wenige Matrixmultiplikationen und eröffnet latenzkritische Mikrocontroller-Implementierungen [31].

Hinsichtlich der Komplexität der ML-Algorithmen wurden NNs und GPs einzeln evaluiert. Es ist hierbei a-priori klar, dass beide Algorithmen schneller und effizienter sind als eine Online-Optimierung, da diese im Grunde nur aus Multiplikation und Addition von Matrizen bestehen. Dies sind für viele Recheneinheiten native Operationen und es können Hardwarebeschleuniger angefertigt werden, um spezielle Abfolgen weiter zu beschleunigen.

Neuronale Netze werden im Allgemeinen als eine Folge von Schichten dargestellt, wobei jede Schicht eine Affine-Nichtlineare-Abbildung realisiert. Eine vollverbundene (feed-forward) Schicht $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ mit Eingabevektor $a \in \mathbb{R}^n$, Gewichtsmatrix $W \in \mathbb{R}^{m \times n}$ und Bias $b \in \mathbb{R}^m$ lässt sich formal als

$$z = W a + b, \quad y = \sigma(z) \quad (2.6)$$

schreiben, wobei σ eine elementweise Aktivierungsfunktion (z. B. ReLU, Sigmoid, Tanh) darstellt. Eine feed-forward-Architektur ist somit die Verkettung solcher Schichten, auch als Tiefe bezeichnet. Für dieses Projekt beschränken wir uns auf diese reinen, dichten, vorwärtsgerichteten Netze. Operatoren wie Batch-Norm, Pooling oder Faltungen werden bewusst ausgeklammert, um einen klaren Fokus auf den elementaren Rechenpfad-Matrix-Vektor-Multiplikation plus punktweise Nichtlinearität zu legen und die nötige Rechenleistung zu minimieren. Jede Schicht besitzt außerdem $m \cdot n + m$ frei trainierbare Parameter, die bei der Inferenz unverändert bleiben und lediglich gelesen werden, wodurch deterministische Antwortzeiten gewährleistet werden.

Das Bilden von $z = W x + b$ erfordert $\mathcal{O}(m n)$ skalare Multiplikationen (zzgl. Additionen). Die elementweise Abbildung $y = \sigma(z)$ verursacht $\mathcal{O}(m)$ elementare Funktionsauswertungen, was praxisnah vernachlässigbar ist. Für die Dimensionsfolge $(n_0, n_1, \dots, n_L)$ der L Schichten ergibt sich $\mathcal{O}(\sum_{\ell=1}^L n_{\ell-1} n_\ell)$ als Gesamtkosten der Vorwärtsauswertung. Im häufigen Fall gleich breiter Schichten $n_\ell \equiv n$ vereinfacht sich dies zu $\mathcal{O}(L n^2)$.

Selbst unter idealisierten Annahmen (lineare Dynamik, quadratische Kosten, reine Box-Constraints) bleibt die bestmögliche asymptotische Komplexität des MPC $\mathcal{O}((n_x + n_u)^3)$, wie oben beschrieben. Im Gegensatz dazu wächst die Inferenzkomplexität eines feed-forward-Netzes linear mit seiner Parameterzahl, erfordert ausschließlich elementare arithmetische Operationen und liefert strikt deterministische Laufzeiten. Dies ist ein entscheidender Vorteil für Echtzeit- und Embedded-Anwendungen, in denen MPC-Solver selbst unter Warmstarts oft an die Grenzen verfügbarer Rechenleistung stoßen.

Gauß-Prozesse können ähnlich wie neuronale Netze in einem Regelungskontext verwendet werden, um komplexe Abbildungen darzustellen, z. B. für Modelle oder Regelgesetze. Formal ist ein Gauß-Prozess (GP) eine Menge von Zufallsvariablen $\{g(a) \mid a \in \mathbb{R}^n\}$, von denen jede endliche Teilmenge eine gemeinsame Gauß-Verteilung besitzt. Zum Beispiel hat die Teilmenge $\{g(x_1), g(x_2), \dots, g(x_N)\}$, die den Eingaben $A = (a_1, \dots, a_N)^\top$ mit $a \in \mathbb{R}^n$ entspricht, eine gemeinsame Gaußverteilung [52]

$$[g(a_1), \dots, g(a_N)]^\top \sim \mathcal{N}(\mu, K),$$

wobei der Mittelwert μ_i durch eine Mittelwertfunktion $m(a)$ gegeben ist, d. h. $\mu_i = m(a_i)$, und die Kovarianzmatrix K über eine Kovarianzfunktion k mit $K_{ij} = k(a_i, a_j)$ definiert ist. Wir schreiben kurz $g(a) \sim \mathcal{GP}(m(a), k(a, a'))$, um zu kennzeichnen, dass g eine Zufallsfunktion ist, die durch einen Gauß-Prozess beschrieben wird [45].

Wir betrachten eine Regressionsaufgabe mit $N \in \mathbb{N}$ Beobachtungen $y(a) = (y(a_1), \dots, y(a_N))^\top$, $A = (a_1, \dots, a_N)^\top$. Die Beobachtungen entstehen aus einer wahren Funktion f mit additivem, homoskedastischem Rauschen

$$y(a) = g(a) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma_n^2), \quad (2.7)$$

wobei σ_n^2 die Rauschvarianz ist. Gesucht ist die Verteilung von $g(A_*)$ an neuen Testpunkten A_* gegeben den Beobachtungen $y(A)$, also $p(g(A_*) \mid y(A))$. Mit

$$g(A_*) \sim \mathcal{N}(m(A_*), K(A_*, A_*)), \quad y(A) \sim \mathcal{N}(m(A), K(A, A) + \sigma_n^2 I_N)$$

ergibt sich die gemeinsame Verteilung

$$\begin{bmatrix} y(A) \\ g(A_*) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(A) \\ m(A_*) \end{bmatrix}, \begin{bmatrix} K(A, A) + \sigma_n^2 I_N & K(A, A_*) \\ K(A, A_*)^\top & K(A_*, A_*) \end{bmatrix}\right).$$

Die bedingte Verteilung an den Testpunkten lautet

$$p(g(A_*) | y(A)) = \mathcal{N}(\mu_*, \Sigma_*),$$

wobei

$$\mu_* = m(A_*) + K(A_*, A) [K(A, A) + \sigma_n^2 I_N]^{-1} (y - m(A)), \quad (2.8)$$

$$\Sigma_* = K(A_*, A_*) - K(A_*, A) [K(A, A) + \sigma_n^2 I_N]^{-1} K(A, A_*)^\top. \quad (2.9)$$

Die Inferenz in (2.8)–(2.9) erfordert die Inversion der $N \times N$ -Kernelmatrix zur Lösung des Gleichungssystems, was in einer naiven Implementation $\mathcal{O}(N^3)$ Rechenzeit und $\mathcal{O}(N^2)$ Speicher kostet. Für einen festen Datensatz kann die Cholesky-Zerlegung $K(A, A) + \sigma_n^2 I_N = LL^\top$ offline berechnet werden. Die anschließende Vorwärts-/Rückwärtssubstitution reduziert die Kosten für einen einzelnen Vorhersagepunkt auf $\mathcal{O}(N)$ für den Mittelwert (2.8) und auf $\mathcal{O}(N^2)$ für die volle Kovarianz (2.9) [49], [56].

Inducing-Point-Methoden wie FITC, VFE oder SVGP wählen $M \ll N$ Stützpunkte $Z = \{z_1, \dots, z_M\}$ und approximieren den GP durch den latenten Vektor $u = g(Z)$. Die Trainingskosten sinken auf $\mathcal{O}(NM^2 + M^3)$, die Speicherkosten auf $\mathcal{O}(NM + M^2)$ und die Vorhersagekosten auf $\mathcal{O}(M^2)$ bzw. $\mathcal{O}(M)$ für Mittelwert und Varianz [29]. Strukturierte Kovarianzen (SKI/KISS-GP) nutzen, dass sich auf regulären Gittern Toeplitz- oder Kronecker-Struktur ergeben. Per schneller Fourier-Transformation oder Woodbury-Identität führt dies zu $\mathcal{O}(N \log N)$ oder sogar $\mathcal{O}(N)$ Trainings- und Vorhersagekosten [56]. Stochastische Gradienten erlauben Mini-Batch-Training großer Datenmengen, da die variationale Evidenzunterschranke (ELBO) sich über Datenpunkte faktorisiert [30].

UAP1.2.TUD.3 (Komplexitätsreduktion MPC) Auch wenn am Ende dieses Vorhabens das Ziel steht, eine klassische MPC durch ein gelerntes NN- oder GP-Modell zu ersetzen, um die beschriebenen Rechenzeitvorteile nutzen zu können, ist es dennoch sinnvoll, sich mit der Rechenzeitreduktion für klassische MPC-Algorithmen zu befassen, da diese notwendig sind, Trainingsdaten für das NN bzw. GP zu erzeugen.

Im Folgenden sind die wichtigsten Methoden vorgestellt, die zur Beschleunigung der Lösung des OCPs einer MPC beitragen können. Diese sind oft problemunabhängig, einige allerdings auch nur auf spezielle Klassen von OCPs (2.4) anwendbar.

Simultaner und Sequenzieller Ansatz Der Weg, das Problem (2.4) direkt mit einem Optimierungsalgorithmus vom Typ Newton zu lösen, wird normalerweise als simultaner Ansatz bezeichnet. Alternativ können die Zustände des Modells als implizite Funktionen von u_k ausgedrückt werden, wenn man die Eindeutigkeit der dynamischen Bedingungen annimmt, wodurch die Gesamtzahl der Variablen in der Formulierung reduziert wird. Da bei diesem Ansatz die Simulations- und Optimierungsteile nacheinander berechnet werden, wird er gemeinhin als sequentiell bezeichnet. Obwohl diese beiden Ansätze mathematisch äquivalent sind, unterscheiden sie sich in ihrer Umsetzung erheblich und erfordern daher unterschiedliche Werkzeuge zur Verbesserung der Rechenleistung. Die von uns gewählten Methoden werden im Folgenden kurz erläutert.

SQP-Formulierung Für SQPs hat sich die Broyden-Fletcher-Goldfarb-Shanno (BFGS)-Methode durchgesetzt, da sie die Inversion der Hesse-Matrix vermeidet und die Komplexität der Newton-Methode von $\mathcal{O}(n^3)$ auf $\mathcal{O}(n^2)$ reduziert [38]. Ein weiterer beliebter Algorithmus ist die Gauss-Newton-Methode mit Beschränkungen, die zwar theoretisch eine lineare Umwandlungsrate hat, aber in der Praxis sehr gut funktioniert [12]. Das adjungierte SQP-Verfahren approximiert zusätzlich die Jakobi-Matrix der KKT-Bedingungen und kann die Gradienten mithilfe der automatischen Differenzierung [27] effizient schätzen.

Kondensation Wenn es sich bei der Dynamik um ein lineares, zeitlich veränderliches, zeitdiskretes System handelt, können wir auf einfache Weise explizite Nebenbedingungen für jeden der Zustände

X in Form der Eingaben U und der Anfangsbedingung x [14] ableiten. Obwohl sich dadurch die Problemgröße reduzieren lässt, kann die resultierende Formulierung nur mit allgemeinen dichten QP-Lösern effizient gelöst werden. Eine Iteration des QP kann in diesem Fall in $\mathcal{O}(N^3 n_u^3)$ [20] gelöst werden.

Ausnutzung der Struktur Im Gegensatz zu den kondensierenden Methoden können wir die ursprünglichen Variablen als Unbekannte behandeln und beobachten, dass die resultierenden KKT-Bedingungsmatrizen fast block-diagonal sind. Dies impliziert, dass sie durch eine Ricatti-Rekursion [50] effizient faktorisiert werden können, was zu QP-Schrittkosten von $\mathcal{O}(N(n_x + n_u)^3)$ führt. Für die Fälle einer großen Horizontlänge N oder einer vergleichbaren Größe von $n_x \approx n_u$ ist diese Methode daher im Allgemeinen der Kondensation vorzuziehen. In der Praxis beanspruchen auch Allzwecklöser wie IPOPT, diese Rechenkomplexität zu erreichen [10], was jedoch wie bereits erwähnt nie formal garantiert wird.

Warmstart Da das MPC-Schema das OCP-Problem (2.4) zu jedem Zeitpunkt löst, können unter der Annahme einer kontinuierlichen Dynamik und einer geringen Modellabweichung die verbleibenden Eingaben der vorherigen MPC-Iteration nahe genug an der Lösung der nächsten Iteration liegen. Es gibt zahlreiche Verfahren, um diese Beobachtung auszunutzen, indem die Anfangsschätzung auf der Grundlage der zuletzt berechneten Lösung ausgewählt wird. In Anlehnung an das Prinzip der dynamischen Programmierung können wir davon ausgehen, dass die Teiltrajektorie einer optimalen Trajektorie optimal bleibt. Wenn also zum Zeitpunkt 0 die optimale Eingabe $(u_0^*, u_1^*, \dots, u_{N-1}^*)$ gefunden wurde, dann ist für einen verkürzten Zeithorizont $(t_1, t_2, \dots, t_N)$ mit der Anfangsbedingung $x_1 = f(x_0, u_0^*)$ auch die Folge $(u_1^*, u_2^*, \dots, u_{N-1}^*)$ optimal sein. Dieses Prinzip ist natürlich nicht strikt anwendbar, wenn der tatsächliche Zustand von der Vorhersage abweicht und wenn der Horizont gleich lang bleiben soll. In einem solchen Fall können wir die Anfangsschätzung für u_N als u_{N-1}^* aus der vorherigen Iteration definieren oder eine stationäre Eingabe für den gewünschten Referenzwert verwenden [21]. In der Praxis hat sich jedoch die einfache Wiederverwendung der letzten optimalen Lösung als Anfangsschätzung für die nächste Iteration bei MPC-Problemen bewährt [13].

Real-time Iterations Wenn wir eine Echtzeit-Implementierung eines MPC-Reglers in Betracht ziehen, wird unsere Berechnungskomplexität durch die Abtastrate der Anlagensensoren eingeschränkt. Da die Lösung von OCP (2.4) eine von null abweichende Zeitspanne in Anspruch nimmt, ist die erzeugte Lösung für den Zustand der Anlage, die die Messungen geliefert hat, veraltet. Diese Herausforderungen werden in der Literatur umfassend untersucht, und es gibt eine Reihe von Ansätzen, um sie zu überwinden. Forscher in [28], [34], [39] stellen verschiedene Ansätze vor, wie die Echtzeit-MPC mit einer einzigen Iteration der zugrunde liegenden iterativen Algorithmen pro Abtastschritt implementiert werden kann.

Anwendungsbeispiel Wir haben uns dafür entschieden, alle oben genannten Ansätze mit Hilfe der Acados-Toolbox [54] zu kombinieren, die ein sehr effizientes Werkzeug für die Implementierung eines rechenintensiven MPC-Reglers mit geringem Speicherbedarf ist, und auf die in Abschnitt 2.1.2 (AP5) entworfene MPC für die Brennstoffzelle anzuwenden. Es war dabei möglich, zu fast identischen Ergebnissen zu kommen, jedoch mit einer deutlichen Reduktion der Rechenzeit.

Reduktion des Eingangsraums Zusätzlich wurde von uns in [6] im Zuge des Projektes durch die Beschränkung des Suchraums auf einen über die Grassmann-Mannigfaltigkeit bestimmten p -dimensionalen Unterraum die Anzahl der Entscheidungsvariablen im OCP drastisch verringert. Anstatt bei jedem Abtastschritt das vollständige, hochdimensionale OCP zu lösen, wird die Optimierung im vorab optimierten niedrigdimensionalen Raum durchgeführt, wodurch sich die Lösungszeit pro Zyklus signifikant reduziert, ohne die Zulässigkeit und Qualität der Steuersequenzen zu beeinträchtigen.

UAP1.2.TUD.4-7 (Effiziente und robuste MPC mit NN/GP Modell) Im Rahmen dieser Arbeitspakete wurde untersucht, wie ML-basierte Methoden die Anwendung von MPC auf eingebetteten Systeme

men vereinfachen können. Der wesentliche Gedanke besteht darin, ein gegebenes, oftmals hochgradig nichtlineares oder hochdimensionales Prozessmodell, wie eine Brennstoffzelle, durch eine Kombination aus einem vereinfachten physikalischen Modell und einem gelernten Korrekturanteil zu ersetzen. Formal schreiben wir

$$f_{\text{complex}}(x,u) \approx f_{\text{reduziert}}(x,u) + f_{\text{ML}}(x,u), \quad (2.10)$$

wobei im Extremfall sogar $f_{\text{reduziert}}(x,u) = 0$ gewählt und somit die gesamte Dynamik gelernt wird, wie in UAP1.2.TUD.2 schon angediskutiert. Es werden dadurch zwei Vorteile erwartet: Das reduzierte Modell ist in der Regel linear oder zumindest von niedriger Ordnung und erlaubt dadurch eine schnelle Prädiktion auf ressourcenbeschränkter Hardware und systematische Modellfehler des reduzierten Modells werden durch f_{KI} kompensiert, sodass die prädiktive Genauigkeit, und damit auch die Robustheit des Reglers, steigt. Für viele reale Anlagen existiert kein Modell, das das Systemverhalten hinreichend exakt beschreibt. Hier setzt der KI-Ansatz an: Auf Basis experimentell aufgenommener Daten lässt sich das hybride Modell (2.10) datengetrieben verfeinern, ohne den Echtzeiteinsatz der MPC zu gefährden.

Zur Illustration diene uns das akademische Beispiel eines invertierten Pendels auf einem Wagen, wie in Abbildung 2.1 dargestellt. Das zugehörige nichtlineare Modell lautet

$$\begin{aligned} \dot{x}_{\text{cart}} &= v, & \dot{v} &= \frac{\mu_1(\theta, \omega) \cos \theta + F + gm \cos \theta \sin \theta}{\mu_2(\theta)}, \\ \dot{\theta} &= \omega, & \dot{\omega} &= \frac{\mu_1(\theta, \omega) + F \cos \theta}{l \mu_2(\theta)}, \end{aligned} \quad (2.11)$$

mit

$$\mu_1(\theta, \omega) = -lm \sin \theta \omega^2, \quad \mu_2(\theta) = M + m(1 - \cos^2 \theta),$$

wobei F die Eingangskraft des Wagens, x_{cart} seine Position, v die Wagensgeschwindigkeit, θ der Pendelwinkel (Null in aufrechter Lage) und ω die Winkelgeschwindigkeit sind. Somit können wir unseren Zustand des Pendels als $x = [x_{\text{cart}}, v, \theta, \omega]^\top \in \mathbb{R}^4$ und die Stellgröße als $u = F \in \mathbb{R}^1$ definieren. Dieses System bildet die experimentelle Basis, um unterschiedliche Lernverfahren für f_{ML} sowie deren Einfluss auf den resultierenden MPC systematisch zu evaluieren.

Zur Identifikation der Korrekturkomponente f_{ML} wurden nach den Maßgaben in UAP.1.2.TUD.3 eine nominelle MPC entworfen und verschiedene Aufschwungvorgänge mit einer Abtastzeit von $\Delta t = 10$ ms simuliert. Insgesamt wurden so $N_{\text{train}} = 5000$ Messpunkte gesammelt. Der so entstehende Datensatz $\mathcal{D} = \{(x_i, u_i)\}_{i=1}^{N_{\text{train}}}$ wurde anschließend zu 80 % für das Training und zu 20 % für die Validierung genutzt.

Als reduziertes Modell $f_{\text{reduziert}}$ wurde hier das um die obere Ruhelage linearisierte Systemmodell verwendet. Für f_{ML} wurde jeweils ein auf GPs und ein auf NNs basierendes Lernmodell untersucht:

- GP-Modell: Für jede Zustandskomponente wurde ein GP mit Radial-Basis-Funktion-Kernel und automatischer Relevanzbestimmung angesetzt.

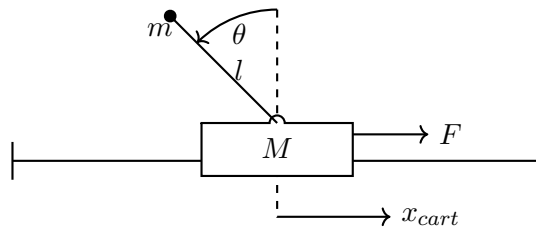


Abbildung 2.1: Invertiertes Pendelsystem. Die Eingangskraft F bewegt den Wagen. Der Ursprung ist für den Winkel θ in der aufrechten Position und für den Wagen x_{cart} in der Mitte der Schiene.

- NN-Modell: Für jede Zustandskomponente wurde ein einzelnes NN (siehe dazu auch unten) mit 1×16 Dichte-Layer und ReLU-Aktivierung angesetzt.

Beide Lernmodelle wurden in einem überwachten Regressions-Setup auf $\mathcal{D}$ trainiert. Hyperparameter (GP-Kernelbreiten, NN-Lernrate und Dropout) wählten wir mittels Bayes'scher Optimierung und Early Stopping verhinderte Overfitting. Die resultierende Regelgüte ist in Abbildung 2.2 zu erkennen.

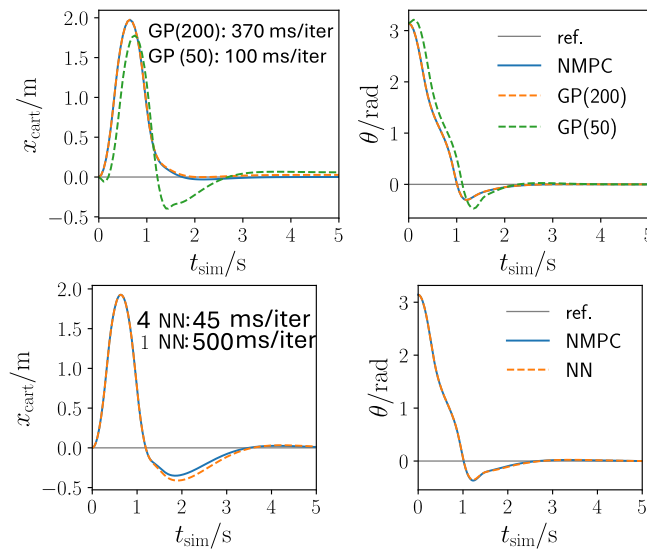


Abbildung 2.2: Vergleich der MPC Varianten für ein invertiertes Pendel mit Gaußprozess (oben) oder neuronales Netz (unten) als hybrides Modell.

Die wesentlichen Beobachtungen aus dieser Studie waren für Gaussprozesse, dass deren Komplexität und Genauigkeit als nichtparametrisches Modell mit der Trainingsdatensatzgröße skaliert. Bei nur 50 Punkten ist noch ein deutlicher Abstand zur Referenz mit perfektem Systemmodell erkennbar; ab etwa 200 Punkten hingegen liegen die Trajektorien nahezu deckungsgleich. Dies wird auch als Fluch der Dimensionalität bezeichnet. Somit muss bei GPs in eingebetteten Anwendungen grundsätzlich eine Balance zwischen Genauigkeit und Recheneffizienz gefunden werden. Für neuronale Netze wird die Komplexität allein durch die Netzarchitektur bestimmt und bleibt unabhängig von der Datenmenge konstant. In unserem Fall erwiesen sich zustand-spezifische Netze numerisch als deutlich stabiler als ein einziges großes Mehrgrößen-Netz. Letzteres führte im Optimalsteuerungsproblem zu längeren Rechenzeiten und vereinzelt Konvergenzproblemen. Die Hybrid-Modelle mit beiden Lernverfahren konnten das Pendel aus zufälligen Startlagen innerhalb von ca. 2,5s aufrichten und stabil halten, ohne die vorgegebenen Kraft- oder Positionsbeschränkungen zu verletzen.

Damit belegt das Experiment, dass bereits wenige hundert Datenpunkte genügen, um das reduzierte physikalische Modell so zu korrigieren, dass die Performance eines ideal modellbasierten MPC praktisch erreicht wird, wobei gleichzeitig ein deutlich geringerer Online-Rechenaufwand auftritt.

Die Erkenntnisse aus diesem UAP wurden als Ansatzpunkte für die Veröffentlichungen [4] und [1] genutzt. Die Robustheit des Reglers mit hybridem Modell kann weiter verbessert werden, wenn Prädiktionsunsicherheiten direkt mit berücksichtigt werden. Zum Beispiel haben wir in [9] die Unsicherheitsschätzungen des Gausprozesses über den Horizont propagiert und über Anpassung der Nebenbedingungen im OCP berücksichtigt.

In [1] schlagen wir einen alternativen Neural-Horizon-MPC (NH-MPC) vor, der ohne Modellvereinfachung auskommt und dennoch die Rechenzeit drastisch reduziert. Die Schlüsselidee ist, die vollständige Systemfunktion $f(x,u)$ nur für die ersten $M < N$ Schritte des Horizonts explizit in das OCP einzubinden; die verbleibenden Zustände $k = M + 1, \dots, N$ werden durch ein offline trainiertes, feed-forward

neuronales Netz $f_s(\cdot)$ prognostiziert:

$$\{\tilde{x}_{M+1}, \dots, \tilde{x}_N\} = f_s(x_M). \quad (2.12)$$

Damit ergibt sich das modifizierte OCP

$$\begin{aligned} \min_{\{x_k\}_0^M, \{u_k\}_0^M} \quad & L(\{x_k\}_0^M, \{u_k\}_0^M) + \tilde{L}(\{\tilde{x}_k\}_{M+1}^{N-1}) + \tilde{V}(\tilde{x}_N) \\ \text{s.t.} \quad & x_0 = x_{\text{init}}, \\ & x_{k+1} = f(x_k, u_k) \quad \forall k \in [0, \dots, M-1], \\ & x_k \in \mathcal{X}_k \quad \forall k \in [0, \dots, M], \\ & u_k \in \mathcal{U}_k \quad \forall k \in [0, \dots, M], \\ & \{\tilde{x}_k\}_{M+1}^N = \tilde{f}_s(x_M), \\ & \tilde{x}_k \in \tilde{\mathcal{X}}_k \quad \forall k \in [M+1, \dots, N] \end{aligned} \quad (2.13)$$

wobei $\{\cdot\}_a^b$ Vektoren mit Werten von a bis b im Horizont darstellen, hierbei sind $a, b \in \mathbb{N}_0$. Des Weiteren stellen $L : \mathbb{R}_x^n \times \mathbb{R}_u^n \rightarrow \mathbb{R}$ die laufenden Kosten der physikalischen Modellzustände und $\tilde{L} : \mathbb{R}_x^n \times \mathbb{R}_u^n \rightarrow \mathbb{R}$ und $\tilde{V} : \mathbb{R}_x^n \rightarrow \mathbb{R}$ die laufenden bzw. End-Kosten der approximierten Zustände dar. Das neuronale Netz $f_s : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_x(N-M)}$ für den in [1] ausgeführten Test besaß drei verborgene Schichten mit je 32 Neuronen und tanh-Aktivierungen. Trainingsdaten entstanden, indem ein Baseline-MPC mit vollem Horizont $N = 30$ optimale Trajektorienpunkte lieferte. Ein approximierter Designfehler von $\gamma = 0,25$ wurde hierbei beachtet, um zu garantieren, dass (2.12) stets innerhalb einer schrumpfenden Mengenhülle $\mathcal{X} \ominus \mathcal{B}_\gamma$ liegt. Unter dieser Voraussetzung kann ([1, Prop. 3.2]) rekursive Lösbarkeit und Nebenbedingungseinhaltung gezeigt werden.

Für das invertierte-Pendel-Beispiel mit $(M, N) = (8, 30)$ ergibt sich gegenüber dem Baseline-MPC (mittlere Solverzeit 160 ± 65 ms, Trajektorienkosten ≈ 105) eine Reduktion der Online-Rechenzeit um $\sim 67\%$ auf 53 ± 17 ms, bei praktisch identischer Regelgüte ($103 \leq J_{\text{NH}} \leq 107$) und ohne Verletzung von Zustands- oder Eingangsbeschränkungen. Im Gegensatz zu rein imitierenden Reglern bleibt also sowohl Performance als auch Sicherheit erhalten.

Zur Berücksichtigung von Modellunsicherheiten in sicherheitskritischen Anwendungen wird ein stochastischer MPC (SMPC) formuliert. Das zugrunde liegende Systemmodell lautet

$$x_{k+1} = f(x_k, u_k) + B(d(x_k, u_k) + w_k), \quad (2.14)$$

wobei $d(\cdot)$ unbekannte, schwer modellierbare Effekte beschreibt und $w_k \sim \mathcal{N}(0, \Sigma_w)$ einen Gauß'schen Prozessrauschterm darstellt. Das SMPC-Optimierungsproblem wird unter probabilistischen Beschränkungen gelöst,

$$\mathbb{P}(x_k \in \mathcal{X}, u_k \in \mathcal{U}) \geq 1 - \varepsilon, \quad k = 0, \dots, N, \quad (2.15)$$

so dass Beschränkungen mit einer kleinen Verletzungswahrscheinlichkeit ε zugelassen werden.

Um den Rauschprozess zu beschreiben, wird ein probabilistisches Modell benötigt. Hierfür werden weiterhin GPs und spezialisierte NNs genutzt:

- **Gaussprozesse** liefern *exakte* Posterior-Varianzen, sind jedoch in $\mathcal{O}(n_{\text{data}}^3)$ skalenbeschränkt.
- **Bayes'sche neuronale Netze** approximieren die Gewichtsverteilungen eines Feed-Forward-Netzes (hier: eine Schicht mit 50 ReLU-Knoten) mithilfe probabilistischer Backpropagation. Die Online-Komplexität hängt dabei nur von der Netzarchitektur ab, nicht aber von n_{data} .

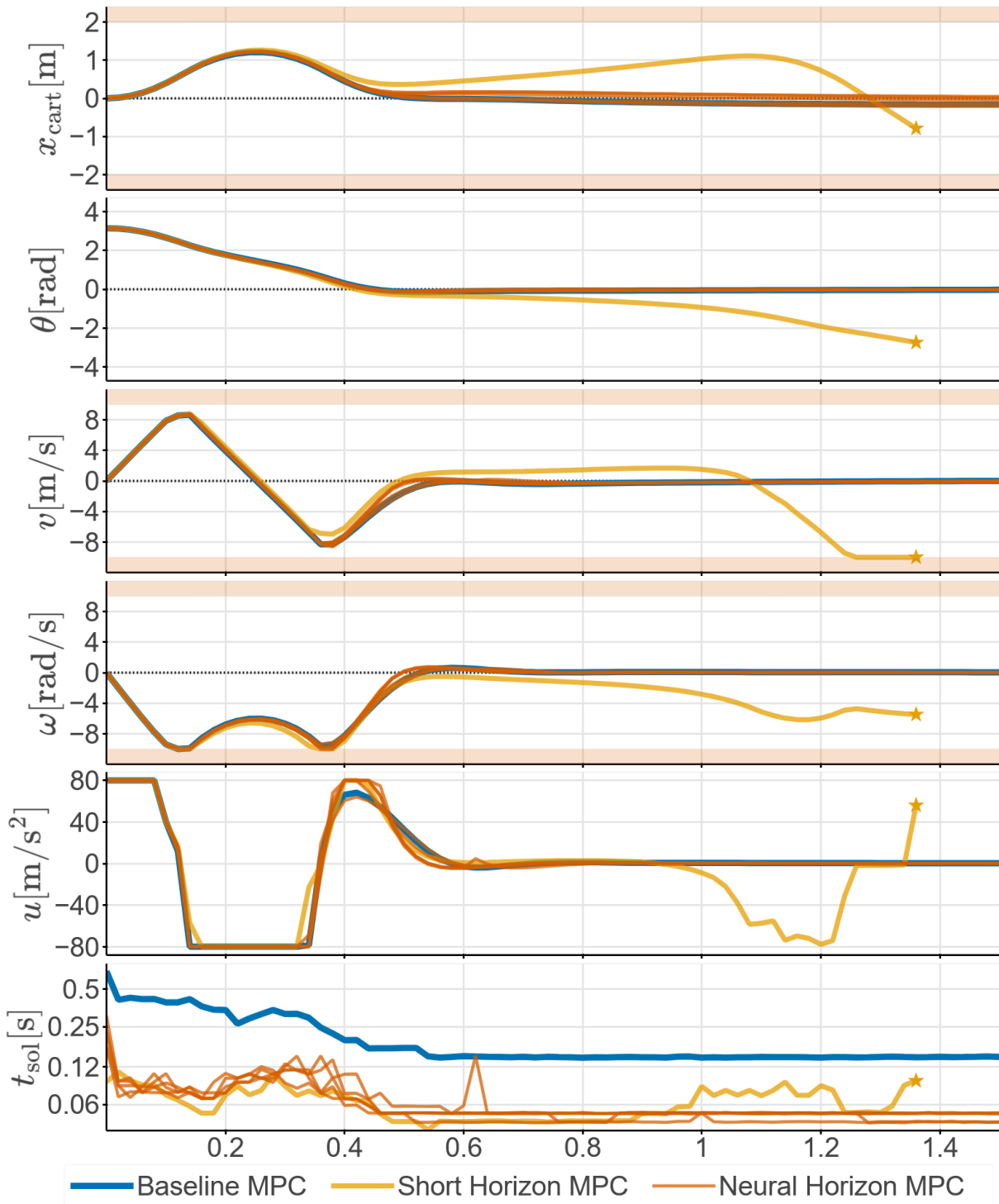


Abbildung 2.3: Aufschwung des umgekehrten Pendels auf einem Wagen. Die Trajektorien sind entlang der horizontalen Achse (in Sekunden) aufgetragen. Der Stern kennzeichnet den Punkt, an dem die MPC mit kurzem Horizont nicht mehr durchführbar ist.

Für diese in [41] gemeinsam mit Kollegen veröffentlichte Studie wurde ein Bioreaktor-Modell genutzt, welches einen realistischeren Rahmen für die nicht-deterministischen Effekte darstellt, die uns interessieren. Dieser Prozess wird durch die zwei Differentialgleichungen

$$\begin{aligned}\dot{X}(t) &= \mu(X, S)X(t) - \frac{F(t)}{V}X(t) - K_d X(t) + w_X(t), \\ \dot{S}(t) &= \frac{F(t)}{V}(S_f(t) - S(t)) - \frac{\mu(X, S)}{Y}X(t) + w_S(t)\end{aligned}$$

beschrieben. Die Zustandsgrößen (X, S) stehen für die Gesamtbiomasse und die Substratkonzentration. Die wachstumsabhängige Biomassezunahme wird durch die Funktion $\mu(X, S)$ modelliert, wohingegen K_d die spezifische Absterberate beschreibt. Zu- und Abfuhr des Mediums erfolgen mit einer Volumenstromrate $F(t)$, bezogen auf das Reaktorvolumen V , wobei das einströmende Substrat dabei die Konzentration $S_f(t)$ besitzt. Der dimensionslose Ausbeutekoeffizient Y verknüpft den Substratverbrauch mit der Biomassebildung. Auf beide Zustände wirken additive, normalverteilte Prozessstörungen, $w_X \sim \mathcal{N}(0, 1)$ und $w_S \sim \mathcal{N}(0, 1)$.

Für dieses Modell wurden 2 800 Datenpunkte generiert, indem der Betrieb mit einer MPC (Abtastzeit $\Delta t = 3$ h, Horizont $N = 80$) simuliert wurde. Das Lernziel war die Modellfehlerabbildung $d = (d_X, d_S)$ und der Datensplit betrug 80:20. Wegen der kubischen Trainingskomplexität musste der GP-Datensatz per Distanzschwelle auf $n_{\text{data}} = 92$ Punkte ausgedünnt werden. Das BNN wurde für beide Zustände separat trainiert (10 Epochen, Gamma-Hyperpriors $\alpha_\lambda = \beta_\lambda = 6$).

Die Evaluation in dieser Studie umfasste folgende Aspekte:

- **Prädiktionsqualität:** Für die Biomasse X erzielten GP und BNN nahezu gleiche Fehler ($\text{RMSE}_X = 0,173$ vs. $0,165$); bei Substratkonzentration S lag das GP leicht vorn ($0,205$ vs. $0,264$).
- **Unsicherheitskalibrierung:** Kalibrationskurven liegen für beide Modelle dicht an der Idealgeraden; die Miskalibrierungs-Fläche beträgt $0,117$ (GP) bzw. $0,107$ (BNN) für X .
- **Simulation der hybriden Modelle im offenen Kreis:** Abbildung 2.4 zeigt Ausschnitte aus den simulierten Sprungantworten für die verschiedenen Modelle, wobei für jede der beiden Zustandsgrößen der jeweils interessantere Bereich dargestellt ist. Hier zeigt sich, insbesondere bei der Substratkonzentration S die deutliche Verbesserung, die mit den auf GP bzw. BNN basierenden hybriden Modellen gegenüber dem nominellen Model, welches die Unsicherheitsinformationen nicht berücksichtigt, erreicht werden.
- **Regelgüte des geschlossenen Kreises:** Abbildung 2.5 zeigt den Verlauf der Biomasse X im geschlossenen Regelkreis. SMPC mit beiden Modellen hält die Biomasse innerhalb der probabilistischen Schranken (2.15), die nominelle Hybrid-MPC ohne Unsicherheitsinformation verletzt diese dagegen sporadisch.

Die Gesamtlaufzeit des SMPC mit GP steigt quadratisch bis kubisch mit der Trainingsdatensatzgröße und überschreitet ab ca. 130 Punkten den konstanten Aufwand des BNN-basierten SMPC (vgl. Fig. 6 in [42]). Damit wird der BNN-Ansatz besonders für große Datenmengen oder Echtzeitsysteme mit strikten Zeitbudgets attraktiv. Bayes'sche neuronale Netze liefern in der betrachteten Fallstudie eine gleichwertige Regelgüte wie Gaußprozesse, bleiben aber skalierbar und erleichtern so bei größeren Datenmengen den Einsatz stochastischer MPC auf leistungsschwacher Embedded-Hardware.

UAP1.2.TUD.8-9 (ML-MPC mit Zustands- und Ausgangsrückführung) Das Ziel dieser Arbeitspakete ist es, bestehende optimierungsbasierte Regler mithilfe von KI-Ansätzen zu lernen und zu approximieren, um den Einsatz auf eingebetteten Steuergeräten zu ermöglichen bzw. zu erleichtern. Der große Vorteil im Vergleich zu dem vorher betrachteten Ansatz mit hybriden Modellen liegt darin,

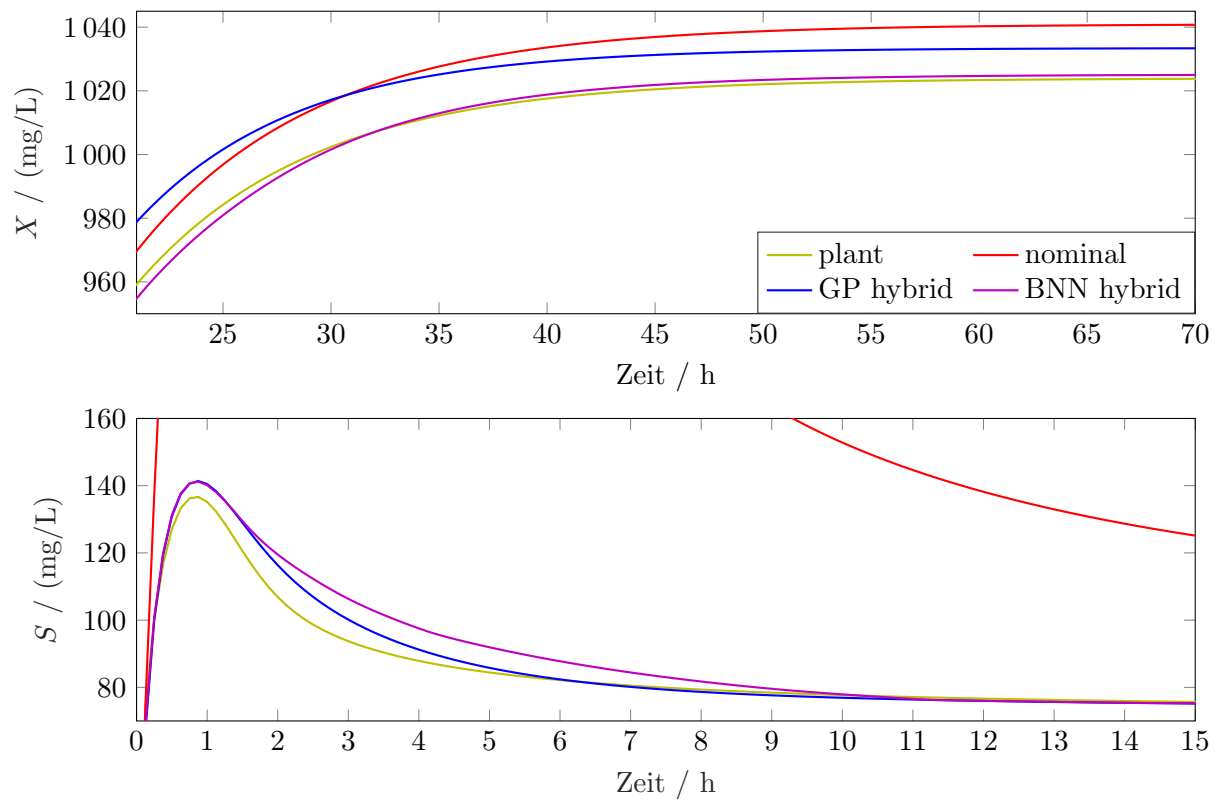


Abbildung 2.4: Auszüge aus Sprungantwort des Anlagenmodells im offenen Regelkreis. [41]

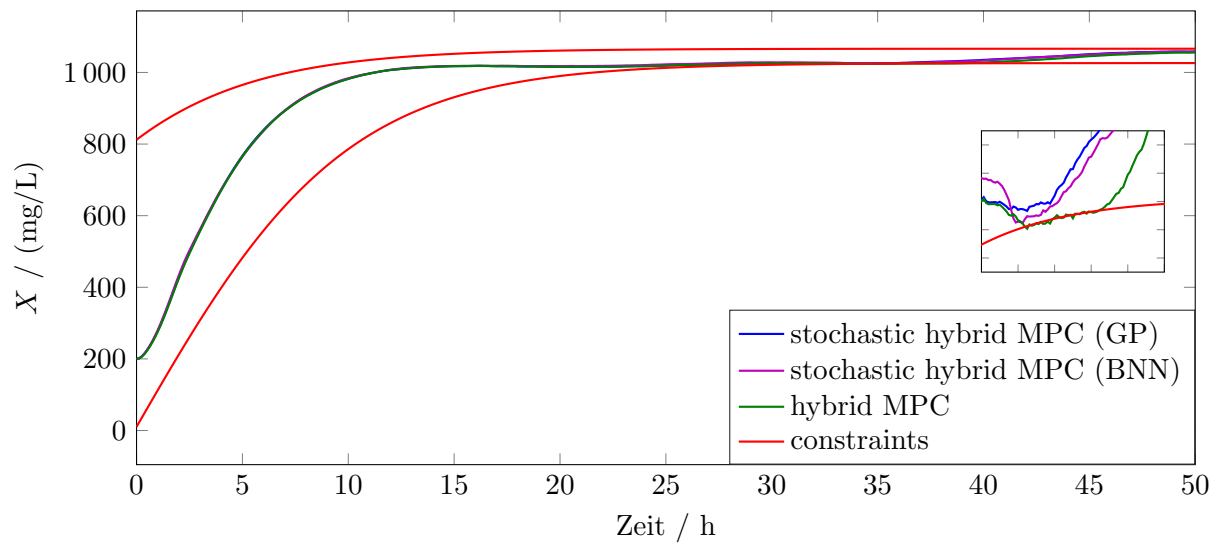


Abbildung 2.5: Simulation der MPCs mit GP und NN im geschlossenen Regelkreis. [41]

dass nun während der Anwendung des Reglers kein Optimierungsproblem gelöst werden muss. Somit kann im Idealfall der Rechenbedarf drastisch reduziert werden.

Auch hier betrachten wir zur Veranschaulichung das invertierte Pendelsystem, und wie in den vorherigen Arbeitspaketen soll der Regler das Pendel von der stabilen unteren Ruhelage in die instabile obere Ruhelage aufschwingen. Wir betrachten im Folgenden zunächst den Fall der vollständigen Zustandsrückführung. Anschließend untersuchen wir die Realisierbarkeit im Fall einer Ausgangsrückführung, bei welcher lediglich auf vergangene Messwerte anstelle des gesamten Zustandsvektors zurückgegriffen werden kann.

Zustandsrückführung Ein häufig auftretender Fall in der Regelungstechnik ist die vollständige Zustandsrückführung, bei der alle Zustandsgrößen des Systems direkt gemessen werden können. In diesem Fall kann das Regelgesetz direkt auf dem aktuellen Zustand des Systems basieren, was eine präzise und schnelle Reaktion auf Systemänderungen ermöglicht. Auch für das Imitationslernen eines MPC-Reglers bildet dieser Zugang eine sinnvolle Grundlage, da die Abbildung vom Zustand auf die Stellgröße ohne zusätzliche Modell- oder Schätzfehler gelernt werden kann. Beim Einsatz von Gauß-Prozessen zur Modellierung dieses Zusammenhangs spielt die Menge an Trainingsdaten eine zentrale Rolle. Die Modellgüte verbessert sich zwar mit wachsendem Datenumfang, gleichzeitig steigen jedoch auch die rechentechnischen Anforderungen. Zunächst haben wir daher untersucht, wie sich diese Anzahl auf die Regelgüte auswirkt. Da die Regelgesetze auch diskontinuierlich sein können, insbesondere bei Systemen mit Eingangsbeschränkungen, ist die Wahl eines Radial-Basis-Funktionen-Kernel hier wenig geeignet. Die Verwendung dieses Kernels führt immer zu unendlich oft differenzierbaren Funktionen. Um Diskontinuitäten besser abbilden zu können, eignet sich zum Beispiel der sogenannte Neuronale-Netz-Kernel

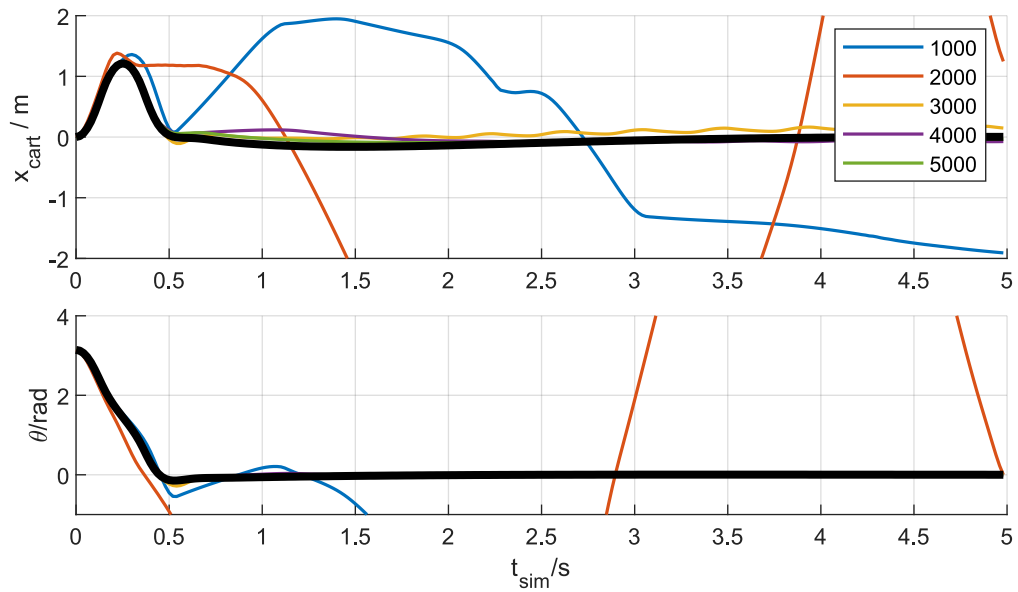
$$c(x, x') = s_f \arcsin \frac{\tilde{x}^T \Lambda \tilde{x}'}{\sqrt{(1 + 2\tilde{x}^T \Lambda \tilde{x})(1 + 2\tilde{x}'^T \Lambda \tilde{x}')}},$$

mit $\tilde{x} = [1 \ x^T]^T$. Im Folgenden verwenden wir daher diese Kernelfunktion. Wie in Abbildung 2.6a zu erkennen, kann der gelernte Regler das Pendel nicht stabilisieren, wenn nur 1 000 oder 2 000 Punkte verwendet wurden, und somit zu wenige Daten vorliegen. Bei 3 000 Datenpunkten stellt sich ein Grenzyklus um die obere Ruhelage ein. Bei mehr als 4 000 Datenpunkten ist der Unterschied zum MPC Regler, welcher durch die schwarze Linie repräsentiert wird, praktisch vernachlässigbar. Der Imitationsregler mit Radial-Basis-Funktionen-Kernel, dessen Ergebnisse in Abbildung 2.6b dargestellt sind, weist bei gleicher Anzahl an Datenpunkten stets eine schlechtere Regelgüte als der Imitationsregler mit Neuronalem-Netz-Kernel auf. Dies unterstützt die vorhergegangene Motivation auf einen Neuronalen-Netz-Kernel zurückzugreifen.

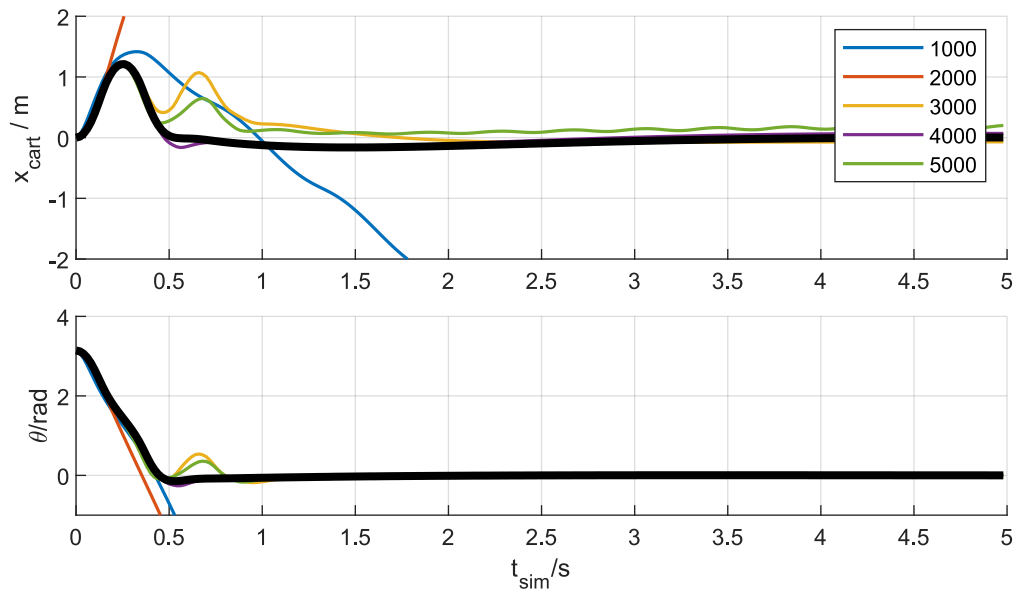
Die benötigte Rechenzeit der gelernten Regler während der Simulation im geschlossenen Regelkreis ist in Abbildung 2.7 veranschaulicht. Die insgesamt höchste Rechenzeit liegt bei ca. 160 μ s für die Auswertung des Reglers. Die Rechenzeit liegt somit deutlich unterhalb der Rechenzeiten mit den hybriden Modellen in Abbildung 2.2, welche bei mehreren Millisekunden liegt.

Bei der Verwendung von neuronalen Netzen ist die Anzahl der Datenpunkte nicht entscheidend für die Komplexität der Online-Auswertung des Netzes. Vielmehr ist hier die Anzahl der Parameter und die Struktur des Netzes relevant. Im betrachteten Beispiel konnte bereits eine sehr gute Regelgüte mit einem Netzwerk aus drei versteckten Schichten mit je 32 Neuronen erreicht werden. Diese war mit der Regelgüte des GP-Imitationsreglers vergleichbar.

Ausgangsrückführung Bisher hatten die gelernten Regler Zugang zum gesamten Zustandsvektor. Häufig kann jedoch nicht der gesamte Zustand online gemessen werden. Im Beispiel des Pendels kann es sein, dass während der online Regelung lediglich die aktuelle Position x_{cart} und der Pendelwinkel θ gemessen werden können und somit der vollständige Zustand, also zusätzlich die aktuelle Geschwindigkeit des Wagens und des Pendels, nicht bekannt ist. Um den Regler dennoch lernen zu können, wurden nun auch vergangene Mess- und Eingangsgrößen als Feature verwendet. Daher nutzen wir im



(a) Neuronales-Netz-Kernel



(b) Radial-Basis-Funktionen-Kernel

Abbildung 2.6: Geschlossener Regelkreis mit GP-Imitationsregler für verschiedene Anzahlen an verwendeten Datenpunkten im GP.

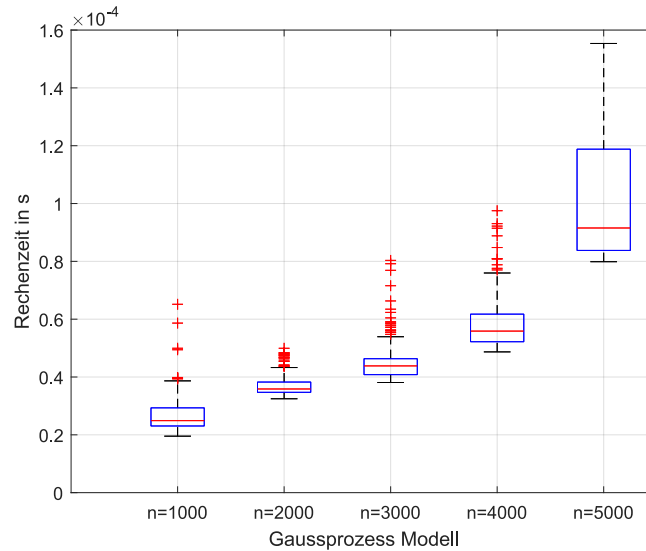


Abbildung 2.7: Rechenzeitstatistik im geschlossenen Regelkreis für den GP-Imitationsregler mit verschiedenen Datenmengen $n = \{1000, \dots, 5000\}$. Die rote Linie kennzeichnet den Median. Die untere bzw. obere Kante der blauen Box gibt das 25 %- bzw. 75 %-Quantil an. Die gestrichelte Linie gibt maximal und minimale Rechenzeiten, ohne Berücksichtigung von Ausreißern. Rechenzeiten, die als Ausreißer betrachtet werden, sind mit einem roten Plus dargestellt und liegen außerhalb des 1,5-fachen Interquartilabstands.

Folgenden den Vektor $\xi_k = [x_{\text{cart},k-1}, \theta_{k-1}, u_{k-1}, x_{\text{cart},k}, \theta_k]^\top$ als Eingang für die gelernten Regler im Zeitschritt k . Die Idee liegt darin, dass zum Beispiel die Information über die aktuelle Geschwindigkeit auch anhand der vergangen und aktuellen Positionsmessung geschätzt werden kann.

Für den GP werden hier nun deutlich mehr Daten benötigt, um eine zufriedenstellende Regelgüte zu erhalten. Genauer gesagt, werden nun 20 000 anstatt 4 000 Punkte benötigt. Der Grund für diese Beobachtung konnte im Rahmen der Projektlaufzeit leider nicht vollständig geklärt werden. Es liegt die Vermutung nahe, dass die Abbildung im Fall der Ausgangsrückführung deutlich komplexer und damit schwieriger zu lernen ist. Wir haben in der Abbildung 2.8 die verschiedenen Regler verglichen. Dort sehen wir, dass der gelernte GP mit Zustandsrückführung und 20 000 Datenpunkten nicht mehr vom nominalen MPC zu unterscheiden ist. Der gelernte Ausgangsrückführungsregler schafft es ebenfalls das Pendel aufzuschwingen, erreicht aber nicht dieselbe Regelgüte wie die Zustandsrückführungsregler.

Zusammenfassend haben wir in diesen Arbeitspaketen die Realisierbarkeit von gelernten Reglern gezeigt. Sie bieten eine starke Vereinfachung des Rechenaufwands im Vergleich zur prädiktiven Regelung bei gleichzeitig nahezu gleichbleibender Regelgüte. In den folgenden Arbeitspaketen, wurden nun die Ideen an dem Anwendungsbeispiel der Brennstoffzelle genutzt.

UAP1.2.TUD.10-13 (Robuste Eingebettete GP bzw. NN) Im Rahmen dieser Arbeitspakete haben wir die Methoden aus den vorherigen Kapiteln auf den Brennstoffzellenstapel angewendet. Das heißt, wir haben sowohl neuronale Netze als auch Gaussprozesse entworfen, welche das Verhalten einer modellprädiktiven Regelung lernen. Details zum verwendeten Modell des Brennstoffzellenstapels haben wir in Abschnitt 2.1.2 beschrieben. Um die Funktion des Stapels zu gewährleisten, ist es wichtig, eine Sauerstoffverarmung der Brennstoffzelle zu vermeiden. Da der Strom I_{st} , welcher durch den Stapel erzeugt wird, direkt mit dem zugehörigen Sauerstoffverbrauch verknüpft ist, ist es nicht immer möglich, mit dem Stapelstrom direkt dem angeforderten Laststrom I_{load} zu folgen, also $I_{\text{st}} = I_{\text{load}}$ zu setzen. Die Hauptaufgabe des Reglers besteht daher darin, den gewünschten Laststrom I_{load} möglichst genau zu verfolgen und gleichzeitig sicherzustellen, dass das Sauerstoffüberschussverhältnis λ_{O_2} sicher über dem Mindestwert bleibt. Hierzu steht die Verdichterspannung als weitere Stellgröße zur Verfügung.

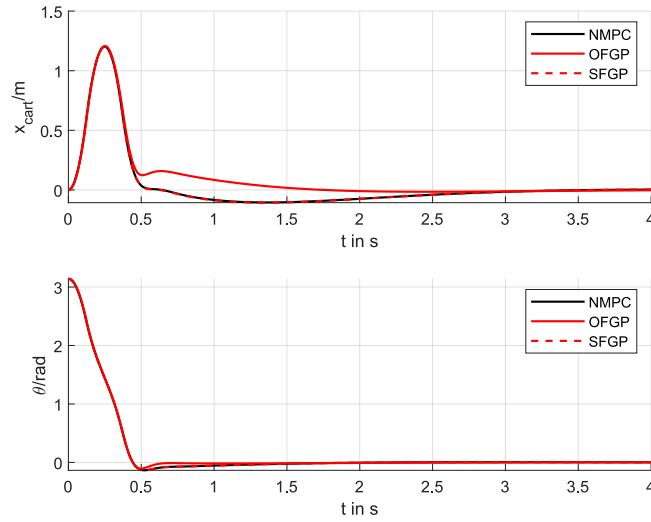


Abbildung 2.8: Auf Ausgangsrückführung ausgelegter GP (OFGP) und auf Zustandsrückführung ausgelegter GP (SFGP), jeweils mit 20 000 Datenpunkten im Vergleich zum nominellen nichtlinearen modellprädiktiven Regler (NMPC).

Konkret nutzen wir daher die Kostenfunktion $J = \sum 0,05(I_{st} - I_{load})^2 + (\lambda_{O_2} - 2)^2$ und die Nebenbedingung $\lambda_{O_2} \geq 1,5$. Für den Verdichter gelten zusätzlich die Bedingungen $20 \text{ krpm} \leq \omega_{cp} \leq 105 \text{ krpm}$ sowie $50V \leq v_{cm} \leq 250V$.

Die Lösung des OCP hängt nicht nur vom Anfangszustand $x(0)$ ab, sondern auch vom geforderten Laststrom I_{load} , sodass die optimale erste Stellgröße nicht nur vom aktuellen Zustand abhängig ist, sondern durch eine Funktion der Form $\hat{u}_{0|k}^* = \text{oep}(x(0), I_{load,k})$ gegeben ist. Daher werden diese Variablen als Eingangsmerkmale

$$\xi = [x^T, I_{load}]^T$$

für die zu lernenden Regler verwendet. Anders als bei den bisher betrachteten Beispielen stehen dem Regler mit dem Stackstrom I_{st} und der Verdichterspannung v_{cp} hier zwei Stellgrößen zur Verfügung. Hier wird dazu so vorgegangen, dass zwei GPs trainiert werden, nämlich eines mit dem Label $\Gamma_1 = u_1 = I_{st}$ und eines mit dem Label $\Gamma_2 = u_2 = v_{cp}$. Für das NN-Ansatz wurde ein einzelnes NN trainiert, welches beide Label $\Gamma = [\Gamma_1, \Gamma_2]$ prädiziert.

Zum Trainieren der Regler wird das OCP offline für 10 000 verschiedene Kombinationen von Anfangsbedingungen und Laststromreferenzen gelöst. Dabei werden die folgenden Wertebereiche abgedeckt: $p_{O_2} \in [0,08, 0,25] \text{ bar}$, $p_{N_2} \in [0,65, 1,75] \text{ bar}$, $\omega_{cp} \in [20, 105] \text{ krpm}$, $p_{sm} \in [1,3, 2,7] \text{ bar}$, $I_{load} \in [80, 340] \text{ A}$. Wenn bestimmte Anfangsbedingungen zu nicht realisierbaren Lösungen führen, da diese beispielsweise zu Verstößen gegen die Mindestanforderung an das Sauerstoffüberschussverhältnis führen bzw. unphysikalisch sind, werden diese verworfen und erneut gesammelt.

Die Datengenerierung erfolgt nicht nur durch zufällige Auswahl von Anfangsbedingungen, sondern auch unter Einbeziehung geschlossener Regelkreissimulationen des modellprädiktiven Reglers. Konkret werden 20 % der Stichproben vollständig zufällig gewählt, während 50 % anhand der Systemtrajektorie im geschlossenen Regelkreis bestimmt werden, wozu zufällige Sollstromverläufe generiert und simuliert wurden. In den verbleibenden 30 % der Fälle wird die nächste Anfangsbedingung ebenfalls aus der geschlossenen Trajektorie abgeleitet, jedoch mit zusätzlichem Rauschen versehen, um die Datenvielfalt zu erhöhen. Wir testen die gelernten Regler mit einem definiertem Laststromprofil.

Anhand der Abbildung 2.9 sehen wir das die gelernten Regler das Verhalten des MPC abbilden können. Insbesondere kann eine Sauerstoffverarmung des Brennstoffzellenstapels verhindert werden.

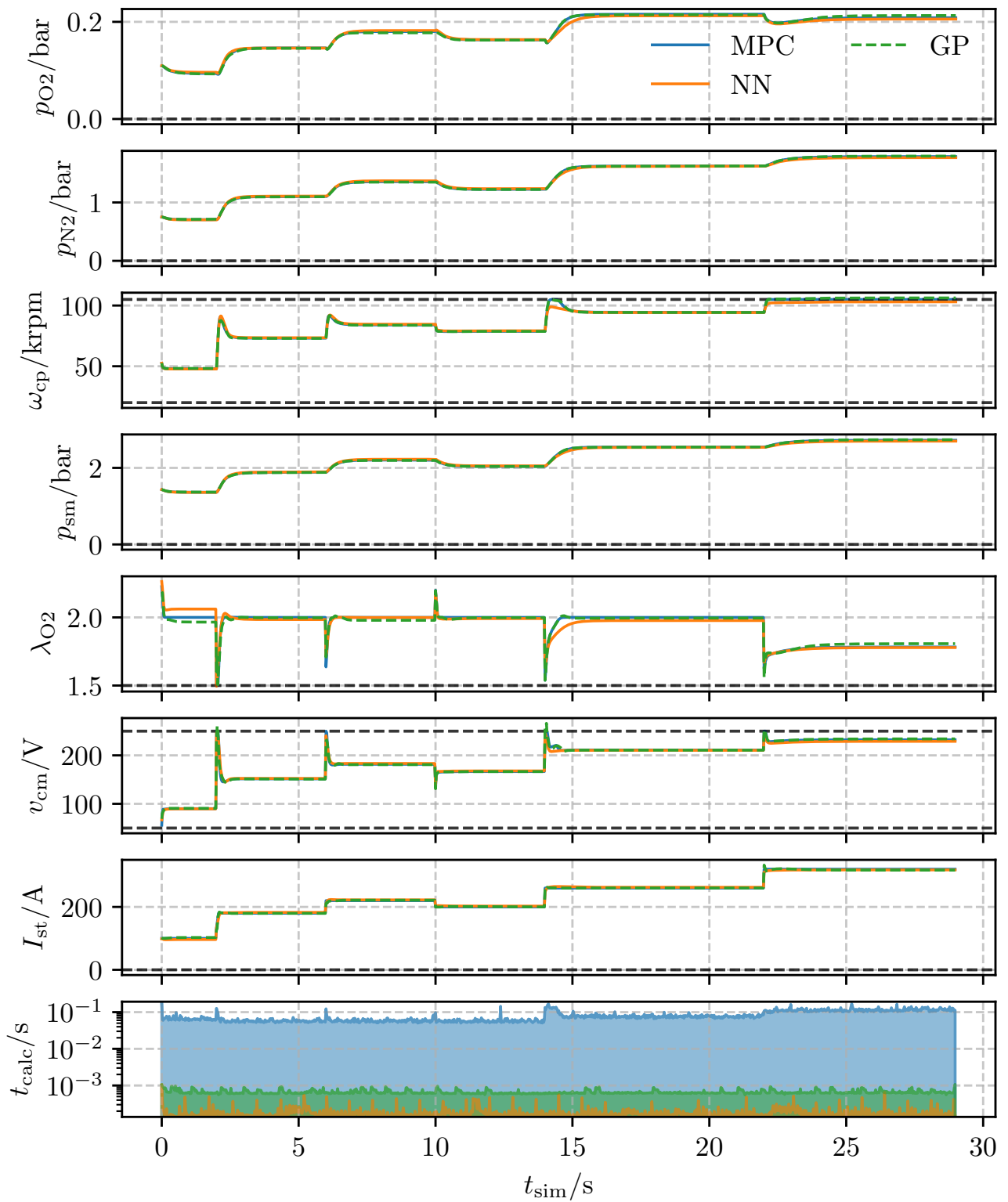


Abbildung 2.9: Vergleich der Imitationsregler (NN und GP) mit dem optimierten MPC. Wertegrenzen sind in schwarz gestrichelt eingezeichnet.

Tabelle 2.1: Rechenzeitstatistik in Millisekunden.

	MPC	GP	NN
Mittelwert	26,6	0,31	0,18
Median	23,5	0,29	0,16
Standardabw.	8,1	0,04	0,06
Minimal	17,0	0,26	0,14
Maximal	71,9	0,52	0,97

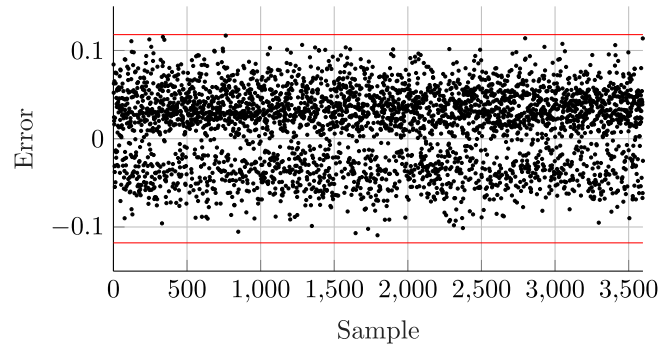


Abbildung 2.10: Approximationsfehler des GP Imitationsreglers für 18000 Szenarien. Die rote Linie kennzeichnet den gewünschten maximalen Fehler. Aus Gründen der Übersichtlichkeit ist nur jedes fünfte Szenario abgebildet.

UAP1.2.TUD.14 (Stabilität und Sicherheit) Das Ziel des Arbeitspakets war es, die Stabilität und Sicherheit von Reglern trotz Rechen- und Speicheroptimierungen zu gewährleisten. Hierzu wurden im Rahmen des Projekts verschiedene Ansätze vorgeschlagen und untersucht.

Szenarioansatz Um zu validieren, dass ein gelernter Regler genau genug ist, um Eingangs-zu-Zustandsstabilität des geschlossenen Kreises zu garantieren, wurde in der im Rahmen dieses Vorhabens entstandene Veröffentlichung [4] auf die Theorie des Szenarioansatzes zurückgegriffen [15]. So reicht es, den Unterschied zwischen den Ergebnissen des Optimalsteuerungsproblems und des gelernten Reglers für einige Zustände zu testen, um mit hoher Sicherheit Rückschlüsse auf die Stabilitätseigenschaften des gelernten Reglers zu ziehen. Insbesondere wurde in [4] das folgende Theorem verwendet: Falls für die Anzahl der Szenarios N_s die Ungleichung $N_s \geq \frac{2}{\varepsilon} \log \frac{1}{\omega}$ für einen gegebenen Risikoparameter $\varepsilon \in (0,1)$ und einen Sicherheitsparameter $\omega \in (0,1)$ erfüllt ist, dann gilt mit hoher Sicherheit $1 - \omega$, dass $P\{\forall x \in \mathcal{X}_{\text{feas}}: \|\kappa_{\text{OCP}}(x) - \kappa_{\text{learned}}(x)\| \leq e_{\text{max}}^*\} \geq 1 - \varepsilon$. Die durch den Szenarioansatz erhaltenen Garantien sind demnach probabilistischer Natur. Der Ansatz ist in Abbildung 2.10 veranschaulicht. Für alle getesteten Szenarios liegt der Approximationsfehler des Imitationsreglers unterhalb gewünschter Schranken. Somit liegt der Fehler mit hoher Wahrscheinlichkeit innerhalb des gewünschten Bereichs.

Als Anschauungsbeispiel haben wir in [4] ein vereinfachtes, mit der Abtastzeit $T_s = 0,1$ diskretisiertes, zweidimensionales Pendelmodell

$$\begin{aligned} x_1(k+1) &= x_1(k) + T_s x_2(k), \\ x_2(k+1) &= x_2(k) + T_s (\sin(x_1(k)) - u), \end{aligned} \tag{2.16}$$

verwendet. Hierbei ist x_1 der Winkel und x_2 die Winkelgeschwindigkeit des Pendels. Für dieses Beispiel haben wir zunächst, durch Verkleinerung der Nebenbedingungen, einen robusten modellprädiktiven Regler ausgelegt. Dabei werden über den Prädiktionshorizont $k = 0, \dots, N$ statt der normalen Zustandsbeschränkungen $x \in \mathcal{X}$ die Beschränkungen $x \in \mathcal{X}_k = \mathcal{X} \ominus B_k$ verwendet. Hierbei ist $B_k = \{x \in \mathbb{R}^{n_x} \mid \|x\| \leq \frac{L_f^k - 1}{L_f - 1} (L_u e_{\text{max}} + \bar{\delta})\}$, wobei L_f die Lipschitzkonstante des Systems ist und in diesem Fall $L_f = 1 + T_s$ beträgt. Die Konstante L_u beschreibt die Auswirkungen des Approximationsfehlers $\|f(x, u + e) - f(x, u)\| \leq L_u \|e\|$ auf die Systemdynamik und ist für das vorliegende Beispiel

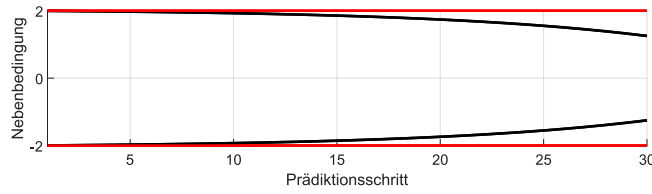


Abbildung 2.11: Verkleinerung der Nebenbedingung über den Prädiktionshorizont in schwarz und tatsächliche Nebenbedingung in rot.

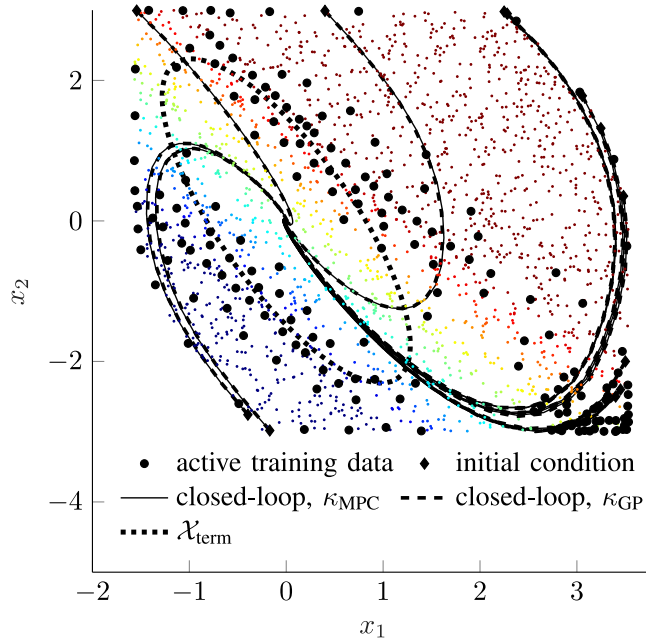


Abbildung 2.12: Trajektorien im geschlossenen Regelkreis mit MPC und GP-Imitationsregler.

als $L_u = T_s$ gegeben. $e_{\max}$ stellt den maximal erlaubten Approximationsfehler und $\bar{\delta}$ eine Schranke für weitere additive Modellfehler dar. Die Verkleinerung der Nebenbedingungen ist in Abbildung 2.11 dargestellt. Der nach diesem Ansatz entworfene GP-Imitationsregler benötigt 178 Datenpunkte, um die gewünschten Fehlerschranken einzuhalten. Die Trajektorien im geschlossenen Regelkreis sind für einige Anfangsbedingungen in der Abbildung 2.12 dargestellt.

Erreichbarkeitsanalyse Die Idee dieses Ansatzes ist es, die erreichbare Menge

$$\mathcal{R}_{x,\tilde{\kappa}}(\mathcal{X}) = \bigcup_{j=0}^{N_{\text{sim}}} \mathcal{R}_{x,\tilde{\kappa}}^j(\mathcal{X}) \quad \text{mit,} \quad \mathcal{R}_{x,\tilde{\kappa}}^j(\mathcal{X}) = \{f_{\tilde{\kappa}}^{\circ j}(x) : x \in \mathcal{X}\},$$

$$\text{wobei} \quad x(k+1) = f(x, \tilde{\kappa}(x)) = f_{\tilde{\kappa}}(x), \text{ sowie} \quad x(k+j) = \underbrace{f_{\tilde{\kappa}} \circ \dots \circ f_{\tilde{\kappa}}}_{j \text{ mal}}(x(k)) = f_{\tilde{\kappa}}^{\circ j}(x(k))$$

des gelernten Reglers zu analysieren, um so gewünschte Eigenschaften des geschlossenen Regelkreises zu validieren. Die erreichbaren Mengen können im Allgemeinen nicht exakt bestimmt werden. Häufig muss daher auf konservative Überapproximationen zurückgegriffen werden. Um dies zu vermeiden, nutzen wir in [5] einen Monte-Carlo-Ansatz zur Bestimmung der erreichbaren Mengen. Die Beschränkung auf erreichbare Mengen erlaubt es, sich beim Reglerentwurf auf die Menge zu beschränken, welche im geschlossenen Regelkreis relevant ist. Für das Aufschwingen des Pendels ist die erreichbare Menge in Abbildung 2.13 mit den gestrichelten Boxen dargestellt. Im Vergleich zum vorherigen Szenarioansatz genügen für den Reglerentwurf bereits die sechs aktiven Datenpunkte, welche in Abbildung 2.13 als schwarze Punkte dargestellt sind, um das Pendel in die obere Ruhelage zu regeln. Folglich ist der entworfene Regler deutlich ressourcenschonender als der mit dem Szenarioansatz entwickelte Regler.

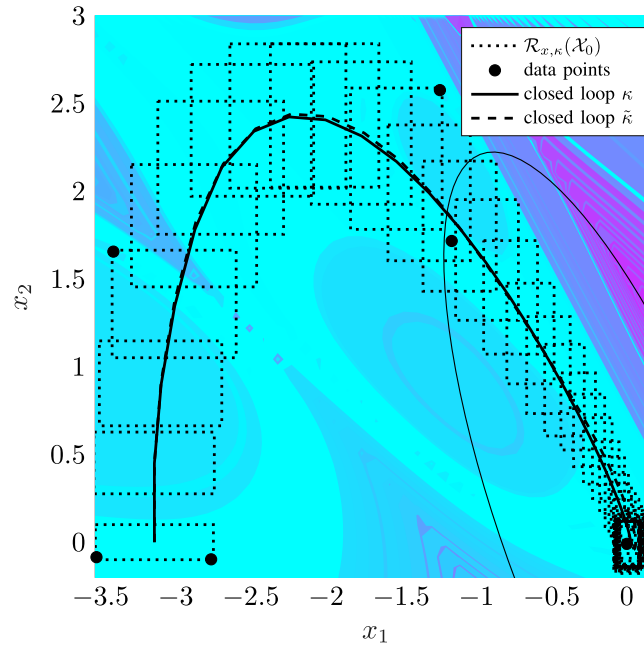


Abbildung 2.13: Aufschwung eines Pendels mit GP-Imitationsregler. Die gestrichelten Boxen beschreiben die geschätzte erreichbare Menge des MPC. Die im Imitationsregler verwendeten Datenpunkte sind mit schwarzen Punkte dargestellt. Die Trajektorien für den Aufschwung in die obere Ruhelage sind für den MPC und Imitationsregler jeweils mit einer durchgezogenen bzw. gestrichelten Linie dargestellt. Die farblichen Konturlinien geben den Approximationsfehler an. Hierbei bedeutet ein dunkleres rot oder dunkleres blau einen größeren Fehler des Imitationsregler.

Außerhalb der betrachteten Mengen verliert der Regler aber, im Gegensatz zum Szenarioansatz, seine Gültigkeit.

Abschätzung des größten Fehlers Die Idee dieses von uns in [2] beschriebenen Ansatzes ist es, anhand der Lipschitzkonstanten des Optimalsteuerungsproblems und der des gelernten Reglers abzuschätzen, wie groß der Fehler zwischen beiden Regler maximal sein kann. Letztendlich ergeben sich daraus Bedingungen, welche angeben, wie dicht der Datensatz sein muss, sodass der gelernte Regler maximal einen Fehler von ε macht. Konkret ist der Fehler $\|\kappa_{\text{OCP}}(x) - \kappa_{\text{KI}}(x)\| \leq \varepsilon \forall x$, falls ein Datensatz gewählt wird, sodass für $\delta = (L_{\text{OCP}} + L_{\text{learned}})^{-1}(\varepsilon - \varepsilon_D)$ $\|x - x_i\| \leq \delta$ erfüllt ist. Hierbei gibt ε_D den Fehler des gelernten Reglers für einen gegebenen Datensatz an. Die hierdurch erhaltenen Garantien sind aufgrund der Abschätzung basierend auf Lipschitzkonstanten deterministischer Natur. Die Idee des Ansatzes ist in Abbildung 2.14 veranschaulicht. Falls der Approximationsfehler an einem Testpunkt x_i kleiner oder gleich ε_D ist, lässt sich der größtmögliche Fehler für umliegende Punkte mit ε bestimmen. Werden also die Datenpunkte dicht genug, so dass $\|x - x_i\| \leq \delta$ gilt, gewählt, kann $\|\kappa_{\text{OCP}}(x) - \kappa_{\text{KI}}(x)\| \leq \varepsilon \forall x$ gewährleistet werden.

Um ein neuronales Netz effizienter zu trainieren wird, in [2] auch vorgeschlagen, die Sensitivitätsinformationen im MPC mit in der Kostenfunktion einzubeziehen. Das bedeutet, neben den standardmäßigen MSE-Kosten

$$\mathcal{L}_{\text{MSE}}(\mathcal{D}, \Theta) = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \|\kappa(x_i) - \kappa_{\text{NN}, \Theta}(x_i)\|^2, \quad (2.17)$$

werden die Kosten

$$\mathcal{L}_{\text{sens}}(\hat{\mathcal{D}}, \Theta) = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \left\| \frac{\partial \kappa(x_i)}{\partial x_i} - \frac{\partial \kappa_{\text{NN}, \Theta}(x_i)}{\partial x_i} \right\|^2, \quad (2.18)$$

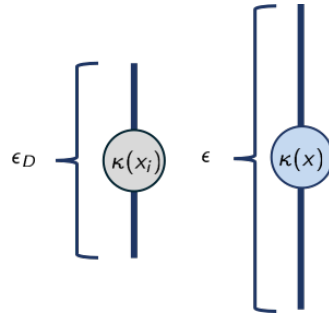


Abbildung 2.14: Idee zur Abschätzung des größtmöglichen Fehlers. Liegt der Fehler an einem Testpunkt x_i innerhalb von ϵ_D kann der Fehler in der Umgebung von x_i mit ϵ abgeschätzt werden.

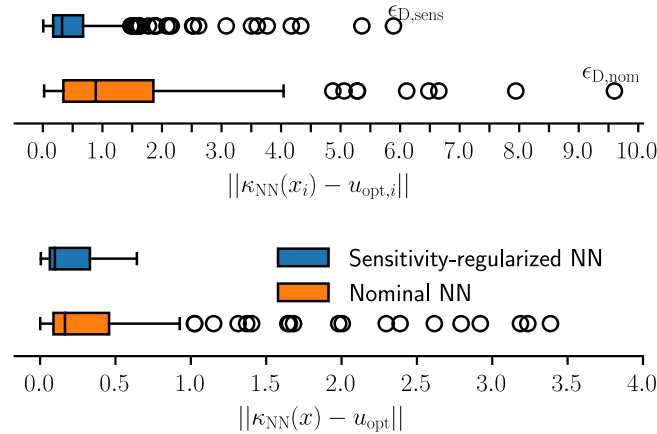


Abbildung 2.15: Vergleich der Approximationsfehler des neuronalen Netzes mit und ohne Berücksichtigen der Sensitivität in der Kostenfunktion.

zu einer Gesamtkostenfunktion

$$\mathcal{L}(\hat{\mathcal{D}}, \Theta) = \lambda_3 \sum_{j=0}^m (\|W_j\|_2^2) + \lambda_1 \mathcal{L}_{\text{MSE}}(\hat{\mathcal{D}}, \Theta) + \lambda_2 \mathcal{L}_{\text{sens}}(\hat{\mathcal{D}}, \Theta), \quad (2.19)$$

hinzugefügt. Hierbei geben λ_i die jeweiligen Gewichtungen an und W_j sind die Wichtungen in der Schicht j des neuronalen Netzes. Die Reduktion von $\|W_j\|_2^2$ führt zu einer Einschränkung der Lipschitzkonstanten des neuronalen Netzes und somit zu einer größeren möglichen Distanz δ . Die Approximationsfehler des mit der angepassten Kostenfunktion entworfenen Netzes sowie des lediglich mit Berücksichtigung der MSE-Kosten entworfenen Netzes sind in Abbildung 2.15 dargestellt. Aus der Abbildung wird deutlich, dass die Einbeziehung der Sensitivität eine deutliche Reduktion des Approximationsfehlers hervorruft.

UAP1.2.TUD.15 (Open-Source-Software) Der für die Ergebnisse des Projekts (AP1 als auch AP5) verwendete Code ist in einer Open-Source-Version auf Github bereitgestellt.

Zum einen ist unter <https://github.com/hilo-mpc/hilo-mpc> die HILO-MPC-Toolbox des Fachgebietes CCPS, die schon vor diesem Projekt begonnen wurde, zu finden. Das Ziel dieser Python-Bibliothek ist es, eine einfache und schnelle Umsetzung von ML-unterstützten optimalen Reglern und Beobachtern zu ermöglichen.

Darüber hinaus sind in diesem Projekt die folgenden Programme und Tools entstanden:

- <https://github.com/Laboratory-CCPS/Neural-Horizon-MPC>
Neural-Horizon MPC
- <https://github.com/Laboratory-CCPS/pem-fc-ampc>
Effiziente Implementierung der MPC für die Brennstoffzelle basierend auf Acados
- <https://github.com/Laboratory-CCPS/pemfc-aimpc>
Modellierung, Beobachterentwurf, MPC-Reglerentwurf und ML-basierte Brennstoffzellenregler

2.1.2 AP5.TUD

UAP5.1.TUD.1 (Modellierung) In einer Brennstoffzelle erfolgt die kontrollierte Knallgasreaktion $2\text{H}_2 + \text{O}_2 \longrightarrow 2\text{H}_2\text{O}$, wobei Energie und Wärme frei werden. An der Anode wird Wasserstoff oxidiert ($2\text{H}_2 \longrightarrow 4\text{H}^+ + 4\text{e}^-$), während an der Kathode Sauerstoff zu Wasser reduziert wird ($\text{O}_2 + 4\text{H}^+ + 4\text{e}^- \longrightarrow 2\text{H}_2\text{O}$). Der Elektronenfluss erzeugt dabei einen elektrischen Strom.

Um dieses Prinzip technisch nutzen zu können, wurden verschiedene Ausführungen der Brennstoffzelle entwickelt. Im Rahmen dieses Projektes wird die Polymerelektrolytbrennstoffzelle (PEMFC – Polymer Electrolyte Fuel Cell) betrachtet. Die dabei in Detail ablaufenden Prozesse sind beispielsweise in [35] beschrieben, und umfassen neben den eigentlichen Oxidations- und Reduktionsreaktionen auch Transportprozesse durch die Membrane und entlang der Gaspfade, den strömungsmechanischen Stofftransport in den Gaspfaden sowie thermische Prozesse. In Abbildung 2.16 ist eine an [35] angelehnte Skizze einer Brennstoffzelle zusammen mit den dort modellierten Eingangsgrößen dargestellt.

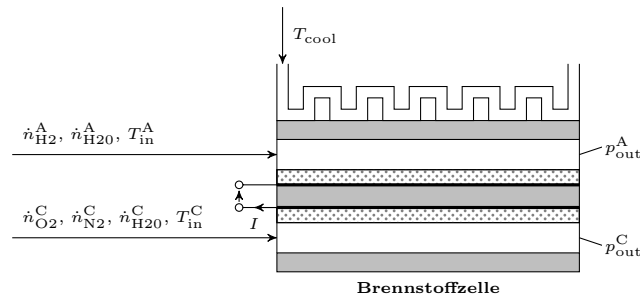


Abbildung 2.16: Brennstoffzelle mit Eingangsgrößen nach der Modellierung in [35]

Für den Einsatz, z. B. in einem Fahrzeug, werden dann viele dieser Brennstoffzellen zu einem Stapel (Stack) kombiniert. Zum Betrieb sind noch zusätzliche Komponenten erforderlich, darunter ein Kühlsystem zur Temperaturkontrolle, ein Wasserstofftank mit Ventilsteuerung sowie ein Kompressor, der Luftzufuhr bereitstellt. Die Umgebungsluft wird durch einen Luftbefeuchter befeuchtet, um die Membran funktionsfähig zu halten [44]. Abbildung 2.17 stellt den Brennstoffzellenstapel mit dieser Peripherie dar. Dabei sind in grau diejenigen Komponenten dargestellt, die bei der unten erläuterten Modellierung nicht genauer betrachtet werden.

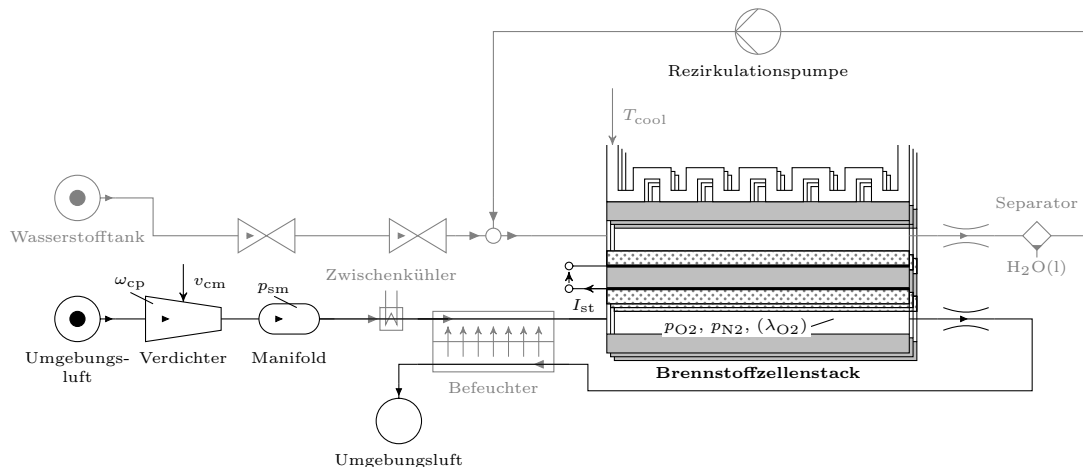


Abbildung 2.17: Brennstoffzellenstapel mit Peripherie

Modellierungsansatz Es wurden verschiedene Modellierungen der PEMFC evaluiert. Insbesondere wurde genauer die Modellierung nach [35] betrachtet, da diese eine für den Projektpartner OvGU

günstige Form besitzt, und es ideal gewesen wäre, eine einzige Darstellungsform innerhalb des Gesamtprojektes zu verwenden. Jedoch hat sich dies aus folgenden Gründen als nicht sinnvoll herausgestellt. Das Modell nach [35] ist ein nichtlineares DAE-Modell, das aus einer Diskretisierung von PDEs entsteht. Die Systemordnung hängt damit von der Anzahl der gewählten örtlichen Diskretisierungsstützstellen ab, wobei die Mindestordnung (bei einem Volumen) 11 ist. Wie oben aufgezeigt beschreibt das Modell nach [35] viele unterschiedliche Teilprozesse (elektro-chemische, strömungsmechanische, thermische), was zu einer starken Streuung der Zeitkonstanten führt. (Bei der Diskretisierung mit nur einem Volumen liegen die Eigenwerte des linearisierten Systems in einem Bereich von ca. -1000 bis $0,01$.) Es ist nicht sinnvoll, einen einzelnen Regler für diese verschiedenen Teilprozesse zu entwerfen.

Daher wurde letztlich für die Modellierung, die in [51] beschrieben ist und auf [44] basiert, gewählt. Dieses betrachtet direkt einen Brennstoffzellenstapel (und keine einzelne Zelle), der schon mit der notwendigen Peripherie für einen Betrieb ausgestattet ist. Es wird hier nur die Regelung des Sauerstoffüberschusses betrachtet. Es wird dazu davon ausgegangen, dass der deutlich schneller reagierende Druck schon unterlagert geregelt wird, und dass die Temperaturregelung in deutlich längeren Zeithorizonten stattfindet.

Mit diesen Annahmen erhält man ein nichtlineares Zustandsraummodell vierter Ordnung mit den Stapelstrom I_{st} und der Verdichterspannung v_{cm} als Eingangsgrößen.

Das Regelziel ist dabei, den geforderten Stapelstrom zu liefern, wobei gewisse Bedingungen, insbesondere ein minimaler Sauerstoffüberschuss, einzuhalten sind.

Modell Das dynamische Verhalten des verwendeten Modells des Brennstoffzellenstapels basiert auf den in [51] beschriebenen Gleichungen und besitzt die vier Zustände

- Partialdruck Sauerstoff p_{O_2} sowie
- Partialdruck Stickstoff p_{N_2} im Kathodenpfad,
- Luftdruck im Volumen nach dem Verdichter p_{sm} und
- Verdichterdrehzahl ω_{cp} ,

die im Zustandsvektor

$$x = [p_{O_2} \quad p_{N_2} \quad p_{sm} \quad \omega_{cp}]^T$$

zusammengefasst sind. Eingangsgrößen sind

- Stapelstrom I_{st} und
- Verdichterspannung v_{vm} ,

die zusammen den Eingangsvektor

$$u = [I_{st} \quad v_{cm}]^T$$

bilden.

Die Dynamik des Sauerstoffpartialdruckes wird durch

$$\dot{p}_{O_2} = \frac{RT_{st}}{M_{O_2}V_{ca}}(W_{O_2,ca,in} - W_{O_2,ca,out} - W_{O_2,react}), \quad (2.20)$$

beschrieben, mit der universellen Gaskonstanten R , der, durch das Kühlsystem konstant gehaltenen, Stapeltemperatur T_{st} , der molaren Masse von Sauerstoff M_{O_2} und dem Stapelvolumen V . Der Term $W_{O_2,ca,in} = \frac{z_{O_2,atm}}{1+w_{atm}}W_{ca,in}$ beschreibt die Sauerstoffzufuhr in die Kathode, wobei z_{O_2} den Masseanteil

von Sauerstoff in der Luft und w_{atm} die relative Feuchtigkeit ist. Der Gesamtfluss in die Kathode $W_{\text{ca,in}} = k_{\text{ca,in}}(p_{\text{sm}} - p_{\text{ca}})$ ist abhängig vom Druckunterschied zwischen der Kathode $p_{\text{ca}} = p_{\text{O}_2} + p_{\text{N}_2} + p_{\text{sat}}$ und dem Zuluftverteiler p_{sm} . Für den Sauerstoffausfluss $W_{\text{O}_2,\text{ca,out}}$ wird eine Düsenströmung angenommen [51]. Die Menge an reagierten Sauerstoff

$$W_{\text{O}_2,\text{react}} = M_{\text{O}_2} \frac{n I_{\text{st}}}{4F} \quad (2.21)$$

ist direkt proportional zum Strom I_{st} , der durch die n Brennstoffzellen fließt. Die Dynamik des Drucks im Zuluftverteiler ergibt sich aus der gewöhnlichen Differenzialgleichung

$$\dot{p}_{\text{sm}} = \frac{RT_{\text{cp}}}{M_{\text{atm}} V_{\text{sm}}} (W_{\text{cp}} - W_{\text{ca,in}}), \quad (2.22)$$

mit dem Luftmassenstrom des Kompressors W_{cp} , der Lufttemperatur des Kompressors T_{cp} und dem Volumen des Zuluftverters V_{sm} . Der Luftmassenstrom des Kompressors wird basierend auf der Kompressordrehzahl ω_{cp} berechnet, wie in [44] beschrieben und hier nicht weiter ausgeführt. Für die Kompressordrehzahl gilt folgende dynamische Gleichung

$$\dot{\omega}_{\text{cp}} = \frac{1}{J_{\text{cp}}} (\tau_{\text{cm}} - \tau_{\text{cp}}), \quad (2.23)$$

mit der Trägheit J_{cp} dem Drehmoment des Motors $\tau_{\text{cm}} = \eta_{\text{cm}} k_t \frac{v_{\text{cm}} - k_v \omega_{\text{cp}}}{R_{\text{cm}}}$ und dem Kompressordrehmoment $\tau_{\text{cp}} = \frac{P_{\text{cp}}}{\omega_{\text{cp}}}$. Hierbei sind k_v , k_t Motorkonstanten, η_{cm} der Wirkungsgrad, v_{cm} die angelegte Spannung und P_{cm} die Leistung des Kompressors. Der Stickstoffpartialdruck kann analog zum Sauerstoffpartialdruck beschrieben werden,

$$\dot{p}_{\text{N}_2} = \frac{RT_{\text{st}}}{M_{\text{N}_2} V_{\text{ca}}} (W_{\text{N}_2,\text{ca,in}} - W_{\text{N}_2,\text{ca,out}}), \quad (2.24)$$

mit der molaren Masse M_{N_2} von Stickstoff.

Eine wesentliche Größe beim Betrieb der Brennstoffzelle ist der Sauerstoffüberschuss

$$\lambda_{\text{O}_2} = \frac{W_{\text{O}_2,\text{ca,in}}}{W_{\text{O}_2,\text{react}}}.$$

Der Nenner gibt den Sauerstoffverbrauch (2.21) an und hängt direkt vom Stapelstrom I_{st} ab. Der Zähler stellt den zugeführten Sauerstoffstrom in die Kathode dar und ergibt sich aus den Zustandsgrößen des Modells.

Zum Entwurf des Beobachters (Soft Sensors) wird auch die Spannung

$$v_{\text{st}} = n \cdot v_{\text{fc}}$$

des Stapels, die sich aus der Hintereinanderschaltung der n Zellspannungen v_{fc} ergibt, als weitere Ausgangsgröße benötigt. Die Zellspannung ergibt sich aus der Differenz der Leerlaufspannung (open-circuit voltage) E und dem Aktivierungsverlust v_{act} , den Ohm'schen Verlusten v_{ohm} sowie den Konzentrationsverlusten v_{conc} ,

$$v_{\text{fc}} = E - v_{\text{act}} - v_{\text{ohm}} - v_{\text{conc}}.$$

Für genauere Informationen zur Berechnung dieser Komponenten siehe [43].

Abbildung 2.18 zeigt den simulierten Verlauf der Zustandsgrößen und des Sauerstoffüberschusses auf einen vorgegebenen Verlauf des Stapelstroms und der Verdichterspannung. Diese Verläufe sind den Beispielen in [43], [51] entnommen.

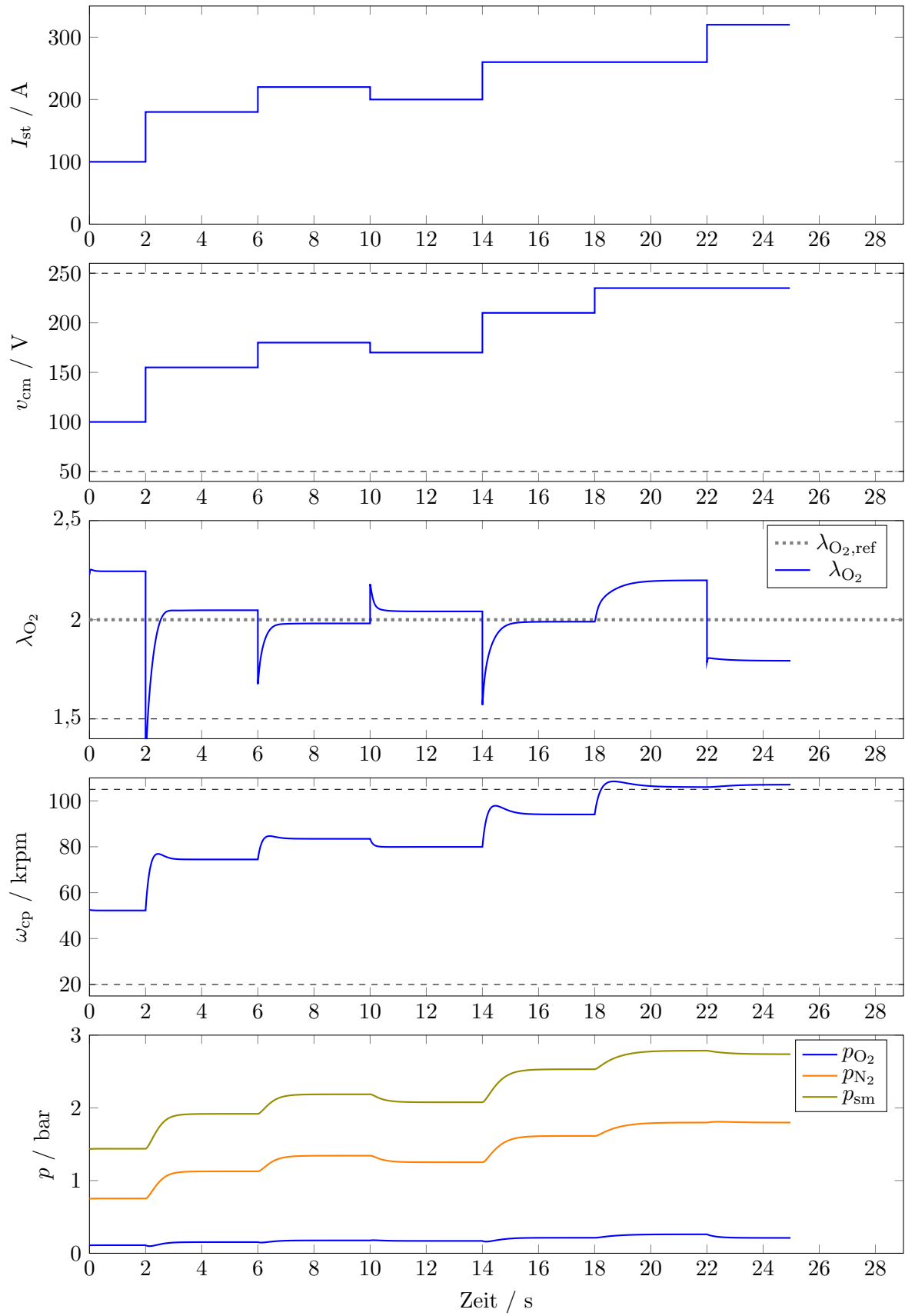


Abbildung 2.18: Simulation des Brennstoffzellenstapels

Zusätzlich zu den Verläufen sind gestrichelt noch die oberen und unteren Grenzen angegeben, die später bei der Regelung berücksichtigt werden. Ebenso ist der Sollwert $\lambda_{\text{O}_2,\text{ref}}$ von 2 durch eine Punktklinie hervorgehoben.

Man erkennt, dass in dem hier vorliegenden rein gesteuerten Betrieb dieser Sollwert nicht dauerhaft erreicht wird, und zudem manche der Grenzen auch über- bzw. unterschritten werden.

Anhand dieser und weiterer Simulationen wurde für jede Eingangs-, Zustands- und Ausgangsgröße auch jeweils ein Intervall bestimmt, in diesem sich diese Größe im Betrieb jeweils bewegt. Damit wurde für jede der Größen eine skalierte Größe abgeleitet, und eine neue Systemdarstellung bezüglich dieser skalierten Größen bestimmt. Bei dieser skalierten Systemdarstellung nimmt jede Eingangs-, Zustands- und Ausgangsgröße im Betrieb nur Werte in einem Intervall von ungefähr -1 bis 1 an. Diese skalierte Darstellung ist aus numerischen Gründen für die Optimierung innerhalb der MPC sowie im Beobachterentwurf vorteilhaft. Die Darstellungen in diesem Bericht verwenden jedoch auch weiterhin immer die physikalischen Größen.

UAP5.2.TUD.1 (Regelziele und Testfall) Das übergeordnete Ziel ist es, die Brennstoffzelle sicher und effizient zu betreiben. Dabei bedeutet „sicher“, dass alle relevanten Größen der Brennstoffzelle in einem Bereich bleiben, der weder übermäßigen Verschleiß oder gar die Gefahr einer sofortigen Schädigung der Brennstoffzelle bedeutet. Dies wären z. B. eine maximale Zelltemperatur, eine maximale Druckdifferenz zwischen den Gaskanälen der Anode und Kathode sowie ein Mindestwert für den Sauerstoffüberschuss.

„Effizient“ bedeutet, dass eine gegebene Leistung mit minimalen Ressourcenverbrauch, d. h. hier minimalen Verbrauch an Wasserstoff, erzeugt wird. Der von der Brennstoffzelle abgegebene Strom ist direkt proportional zu dem verbrauchten Wasserstoff, sofern angenommen wird, dass unverbrauchter Wasserstoff, wie in Abbildung 2.17 eingezeichnet, zurückgeführt und Leckage vernachlässigt wird. Betrachtet man nur die Brennstoffzelle an sich, würde der maximale Wirkungsgrad daher dann erreicht werden, wenn ein gegebener Strom mit der maximal möglichen Spannung erzeugt wird. Hierbei würde aber der Energiebedarf der gesamten Peripherie, die zum Betrieb der Brennstoffzelle notwendig ist, vernachlässigt werden.

Das andere Extrem der Effizienzbetrachtung stellt die Berücksichtigung des Gesamtsystems der Stromversorgung eines mit einer Brennstoffzelle ausgestatteten Fahrzeuges dar, das auch die Batterie und mögliche andere Zwischenspeicher für die Energie berücksichtigt.

Die gewählte Betrachtung hängt auch von der Ebene der Regelung ab, die hier betrachtet werden soll. So treten in einer Brennstoffzelle viele unterschiedliche Prozesse mit einer breiten Streuung der Zeitkonstanten auf, von sehr schnellen elektrischen Abläufen, die im Bereich von Millisekunden stattfinden, über die strömungsmechanische Vorgänge, die im Bereich einer Sekunde stattfinden, hin zu thermischen Effekten, die über mehrere Minuten wirken. Es ist nicht sinnvoll, hier eine einzelne Regelung zu entwerfen, die alle diese Prozesse gemeinsam betrachtet.

Es wurde sich letztlich entschieden, die Regelung des abgegebenen Stromes sowie die Regelung des Luftmassenstroms im Kathodenpfad zu betrachten. Die Aufgabe der Regelung ist es, den tatsächlich von der Brennstoffzelle gelieferten Strom I_{st} möglichst genau an einen von einer übergeordneten Einheit angeforderten Sollstrom I_{load} anzupassen, und dazu den notwendigen Luftstrom bereitzustellen, wozu als Stellglied ein Verdichter zur Verfügung steht.

Damit werden folgende Annahmen getroffen, die auch schon in dem oben beschriebenen Modell berücksichtigt sind. So wird davon ausgegangen, dass ein Umrichter vorhanden ist, der in der Lage ist, einen gewünschten Strom I_{st} unmittelbar (aus Sicht der hier betrachteten Regelung) einzustellen und abzunehmen. Ebenso wird davon ausgegangen, dass die Druckregelung im Anodenpfad (Wasserstoff)

schneller als die hier betrachtete Regelung im Kathodenpfad ist, so dass diese hier nicht betrachtet werden muss, sondern davon ausgegangen werden kann, dass immer genügend Wasserstoff zur Verfügung steht und die Grenzen der Druckdifferenz über die Membran eingehalten werden. Auf der anderen Seite betreffen die Befeuchtung der Luft und die Temperaturregelung der Zelle deutlich langsamere Prozesse, die ebenfalls an anderer Stelle geregelt werden. Diese hier nicht direkt betrachteten Komponenten sind in der Darstellung in Abbildung 2.17 in Grau dargestellt.

Damit fällt von den oben genannten Faktoren für einen „sicheren“ Betrieb die Sicherstellung eines Mindestsauerstoffüberschusses von $\lambda_{\text{O}_2, \text{min}} = 1,5$ in den Aufgabenbereich der zu entwerfenden Regelung. Ebenso muss auch sichergestellt werden, dass der notwendige Verdichter nicht zu schnell, aber auch nicht zu langsam dreht, d. h. die Verdichterdrehzahl soll während des Betriebs zwischen 20 krpm und 105 krpm liegen. Zuletzt ist zu berücksichtigen, dass die Stellgröße der Verdichterspannung ebenfalls nicht beliebig gewählt werden kann. So ist die maximale Spannung durch die verfügbare Spannung auf 250 V begrenzt und es sollte immer eine gewisse Spannung anliegen, so dass hier die untere Grenze nicht auf den physikalisch minimalen Wert 0 V gegeben ist, sondern ein unterer Grenzwert von 50 V verwendet wird.

Hinsichtlich der Effizienz ist auf der Ebene unserer Betrachtung die Energieaufnahme des Verdichters zu berücksichtigen. Der Wirkungsgrad, d. h. das Verhältnis von eingesetztem Wasserstoffstrom und der Netto-Leistungsentnahme (an den Klemmen entnommene elektrische Leistung abzüglich des Leistungsbedarfs des Verdichters) könnte dazu direkt in das Gütemaß der MPC aufgenommen werden. Dies wurde auch implementiert und untersucht, jedoch erhöht dies die Komplexität des Optimierungsproblems deutlich. Auch zeigt sich, dass ein fester Wert des Sauerstoffüberschusses von $\lambda_{\text{O}_2, \text{ref}} = 2$ zu einem Betrieb bei hoher Effizienz führt, da auch die Kennlinien, die für einen stationären Zustand den optimalen Sauerstoffüberschuss angeben, relativ flach verlaufen.¹

Als Testfall dienen Folgen von Sprüngen des Sollstroms I_{load} . Für die in diesem Bericht gezeigten Darstellungen wurde dafür der Verlauf des Stapelstroms aus Abbildung 2.18 verwendet. Bewertet wird das Einhalten der gegebenen Grenzen sowie das Abweichungen des Sollstromes und des Sauerstoffüberschusses zu dem jeweils gewünschten Wert.

UAP5.2.TUD.2 (Entwurf Soft-Sensor/Beobachter) Die MPC wird hier als Zustandsregelung entworfen werden, d. h. es wird davon ausgegangen, dass zu jedem Rechenzeitpunkt der MPC der aktuelle Zustand bekannt ist. Hierzu müssen die Zustände entweder direkt messbar sein, oder diese müssen anhand des Verlaufs von Messungen anderer Systemgrößen geschätzt werden können.

Von den vier Zustandsgrößen kann bei der Kompressordrehzahl ω_{cp} und dem Verteilerdruck p_{sm} davon ausgegangen werden, dass diese mit vertretbarem Aufwand gemessen werden können, wobei beim Druck von höheren Unsicherheiten auszugehen ist. Die Partialdrücke von Sauerstoff und Stickstoff im Kathodenkanal sind dagegen prinzipbedingt nicht messbar. Theoretisch könnte man den Gesamtdruck im Kathodenkanal messen, und müsste dann aber noch das Verhältnis von Sauerstoff und Stickstoff bestimmen, um auf die Partialdrücke schließen zu können. Dies erscheint für eine technische Umsetzung unrealistisch.

Mit den beiden Messgrößen ω_{cp} und p_{sm} alleine können die Partialdrücke jedoch nicht rekonstruiert werden, wie eine Beobachtbarkeitsanalyse, in Einklang mit [43], zeigt. Eine mögliche weitere Messgröße, die die Beobachtbarkeit sicherstellt, und die auch praktisch gut messbar ist, ist die Stapelspannung v_{st} .

Eine Alternative zu der Kombination einer MPC mit Zustandsrückführung und Zustandsschätzer wäre es auch, die MPC direkt als Ausgangsregelung zu entwerfen. Dies wurde in Erwägung gezogen, jedoch

¹Würde man dies noch weiter optimieren wollen, erscheint es auch günstiger, den Sollwert für den Sauerstoffüberschuss aus Kennfeldern abzulesen und diesen der Regelung vorzugeben.

hat eine Analyse der verschiedenen Optionen gezeigt, dass das hier gewählte Vorgehen aussichtsreicher ist.

Ein klassischer Zustandsbeobachter ist das Kalman-Filter, welches auf stochastischen Überlegungen basiert und neben den Zuständen auch Schätzungen für die Kovarianzmatrix der geschätzten Zustände liefert. Das Kalman-Filter kann nicht hier nicht in der Grundform verwendet werden, da das vorliegende System nichtlinear ist. Zwei Varianten des Kalman-Filters für nichtlineare Systeme sind das erweiterte Kalman-Filter (Extended Kalman Filter) und das „Unscented Kalman Filter“, siehe z. B. [48]. Ersteres prädiziert die Kovarianzen anhand einer Linearisierung der Systemdarstellung, während die zweite Variante die Systemgleichung über einen Zeitschritt mehrfach auswertet, und aus der Streuung der Ergebnisse die Unsicherheiten ermittelt. Für das erweiterte Kalman-Filter dürfen die Nichtlinearitäten nicht zu stark sein, damit die Linearisierung noch sinnvolle Ergebnisse liefert. Für das vorliegende Problem hat sich gezeigt, dass auch das erweiterte Kalman-Filter anwendbar ist. Für eine konkrete Implementierung wäre dann abzuwägen, welche Darstellungsform sich auf dem Zielsystem schneller berechnen lässt.

Es soll noch erwähnt sein, dass man einen Zustandsbeobachter auch als Moving-Horizon-Schätzer (MHE, moving horizon estimator) entwerfen könnte, welches ähnlich einer MPC aufgebaut ist und auf einer wiederholten Lösung eines Optimierungsproblems basiert [46]. Der dadurch entstehende Mehraufwand bei der Berechnung scheint hier aber nach den mit den Kalman-Filtern erreichten Ergebnissen nicht gerechtfertigt. Auch wäre es viel aufwändiger, dieses durch ein NN oder GP zu ersetzen, da die Anzahl der Eingangsgrößen in den MHE deutlich größer als die Anzahl der Eingangsgrößen bei der MPC ist.

Im Folgenden ist ein Ergebnis des erweiterten Kalman-Filters vorgestellt.

Erweitertes Kalman-Filter Das erweiterte Kalman-Filter wird hier in der „hybriden“ Form implementiert, siehe z. B. [48]. D. h. dass im Prädiktionsschritt die Systemgleichungen kontinuierlich betrachtet werden, während die Korrekturschritte nur zu diskreten Zeitpunkten stattfinden.

Das Filter geht damit von den Systemgleichungen

$$\begin{aligned}\dot{\underline{\mathbf{x}}}(t) &= \mathbf{f}(\underline{\mathbf{x}}(t), \mathbf{u}(t)) + \underline{\mathbf{n}}_x(t) \\ \tilde{\mathbf{y}}_k &= \mathbf{h}(\underline{\mathbf{x}}_k, \mathbf{u}_k) + \underline{\mathbf{n}}_{y,k}\end{aligned}$$

aus, wobei der Index k für den jeweiligen Wert zum Zeitpunkt t_k steht, z. B. $\mathbf{u}_k = \mathbf{u}(t_k)$. Dabei stellen $\underline{\mathbf{n}}_x(t)$ und $\underline{\mathbf{n}}_{y,k}$ weiße Rauschprozesse mit den Kovarianzen $E(\underline{\mathbf{n}}_x(t)\underline{\mathbf{n}}_x^T(t')) = \mathbf{Q}(t) \cdot \delta(t-t')$ bzw. $E(\underline{\mathbf{n}}_{y,k}) = \mathbf{R}_k$ dar, die das Zustands- bzw. Messrauschen modellieren. Alle unterstrichene Größen sind in diesem Abschnitt stochastische Signale.

Die Prädiktion von einem Abtastschritt zum nächsten erfolgt über die Simulation der nichtlinearen Systeme

$$\begin{aligned}\dot{\hat{\underline{\mathbf{x}}}}(t) &= \mathbf{f}(\hat{\underline{\mathbf{x}}}(t), \mathbf{u}(t)) & \hat{\underline{\mathbf{x}}}(t_k) &= \hat{\underline{\mathbf{x}}}_{k,+} \\ \dot{\mathbf{P}}(t) &= \mathbf{A}(t)\mathbf{P}(t) + \mathbf{P}(t)\mathbf{A}^T(t) + \mathbf{Q}(t) & \mathbf{P}(t_k) &= \mathbf{P}_{k,+} .\end{aligned}$$

von t_k bis zum Zeitpunkt t_{k+1} . Diese Größen zum Zeitpunkt t_{k+1} entsprechen dann den neuen Schätzwerten des Kalman-Filters,

$$\begin{aligned}\hat{\underline{\mathbf{x}}}_{k+1,-} &= \hat{\underline{\mathbf{x}}}(t_{k+1}) \\ \mathbf{P}_{k+1,-} &= \mathbf{P}(t_{k+1}) .\end{aligned}$$

Dabei ist

$$\mathbf{A}(t) = \left. \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}} \right|_{\hat{\underline{\mathbf{x}}}(t), \mathbf{u}(t)}$$

die linearisierte Zustandsmatrix. Für die vorliegende Anwendung ist $\mathbf{u}(t)$ während eines Zeitschrittes konstant, da der Regler die Stellgrößen nur zu den Abtastzeitpunkten ändert. Es hat sich darüber hinaus auch gezeigt, dass es ausreichend ist, je Zeitschritt die Zustandsmatrix nur einmal, für den Wert des geschätzten Zustandes bei t_k , zu linearisieren und diesen Wert für den gesamten Zeitschritt zu bestimmen. Auch gehen wir davon aus, dass die Kovarianzmatrizen über einen Zeitschritt ebenfalls als näherungsweise konstant angenommen werden können. Damit vereinfacht sich die Prädiktion zu

$$\dot{\hat{\mathbf{x}}}(t) = \mathbf{f}(\hat{\mathbf{x}}(t), \mathbf{u}_k) \quad \hat{\mathbf{x}}(t_k) = \hat{\mathbf{x}}_{k,+} \quad (2.25)$$

$$\dot{\mathbf{P}}(t) = \mathbf{A}_k \mathbf{P}(t) + \mathbf{P}(t) \mathbf{A}_k^T + \mathbf{Q}_k \quad \mathbf{P}(t) = \mathbf{P}_{k,+} \quad (2.26)$$

mit

$$\mathbf{A}_k = \left. \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}} \right|_{\hat{\mathbf{x}}_k, \mathbf{u}_k}.$$

Beim Eintreffen neuer Messungen, hier mit der Abtastzeit T_s , findet eine Korrektur der prädizierten Schätzwerte statt. Diese erfolgt mit

$$\begin{aligned} \mathbf{K}_k &= \mathbf{P}_{k,-} \mathbf{C}_k^T \cdot (\mathbf{C}_k \mathbf{P}_{k,-} \mathbf{C}_k^T + \mathbf{R}_k)^{-1} \\ \hat{\mathbf{x}}_{k,+} &= \hat{\mathbf{x}}_{k,-} + \mathbf{K}_k \cdot (\tilde{\mathbf{y}}_k - \mathbf{h}(\hat{\mathbf{x}}_{k,-}, \mathbf{u}_k)) \\ \mathbf{P}_{k,+} &= (\mathbf{I} - \mathbf{K}_k \mathbf{C}_k) \cdot \mathbf{P}_{k,-} \end{aligned}$$

wobei $\mathbf{C}_k$ die linearisierte Ausgangsgleichung des Systems ist,

$$\mathbf{C}_k = \left. \frac{\partial \mathbf{h}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}} \right|_{\hat{\mathbf{x}}_{k,-}, \mathbf{u}_k}.$$

Beispiel Abbildung 2.19 zeigt beispielhaft eine Simulation des Kalmanfilters. Das Kalmanfilter ist hierzu anhand der skalierten Modelldarstellung implementiert. Die Systemgleichung (2.25) als auch der Differenzialgleichung (2.26) der Kovarianzmatrix sind hier beide mit dem expliziten Eulerverfahren diskretisiert, wobei je Rechenschritt des Kalmanfilters von $T_s = 0,025\text{ s}$ vier Simulationsschritte durchgeführt wurden.

Für dieses Beispiel sind die Verläufe der oben vorgestellten Simulation aus Abbildung 2.18 verwendet. Auf die Messgrößen wurde weißes Rauschen addiert. Die verrauschten Messsignale sind in den entsprechenden Plots von Abbildung 2.19 in Grau dargestellt. Dabei betrugen die Standardabweichungen des Rauschens bei ω_{cp} ca. 1 %, bei v_{st} ca. 2 % und bei p_{sm} ca. 10 % des jeweiligen Arbeitsbereiches. Für die Startwerte der geschätzten Zustände wurden für diese Simulation die wahren Startzustände $p_{\text{O}_2}(0)$, $p_{\text{N}_2}(0)$, $\omega_{\text{cp}}(0)$ und $p_{\text{sm}}(0)$ mit den Faktoren 2, 0,25, 1,02, und 1,2 multipliziert. Tatsächlich sind diese starken Abweichungen im Initialwert der Partialdrücke unnötig groß, da sich hier über den Gesamtdruck p_{sm} im Verteiler sowie der ungefähr bekannten Zusammensetzung der Luft bessere Startwerte abschätzen lassen, jedoch zeigt dies hier, dass der Filter auch bei größeren Fehlern noch konvergieren kann.

Zusätzlich wurde für das im Filter verwendete Modell noch der Kompressorwirkungsgrad mit 0,9 multipliziert, um eine Modellabweichung zu simulieren.

Die geschätzten Verläufe der Zustände sind in Abbildung 2.19 in Orange dargestellt. Man erkennt zum einen, dass der falsche Startwert sehr schnell korrigiert wird. Zum anderen ist im weiteren Verlauf zwar die Auswirkung des Rauschens auf die Schätzung deutlich zu erkennen, jedoch wird den wahren Verläufen insgesamt trotz des hohen Rauschens und der Modellabweichung sehr gut gefolgt.

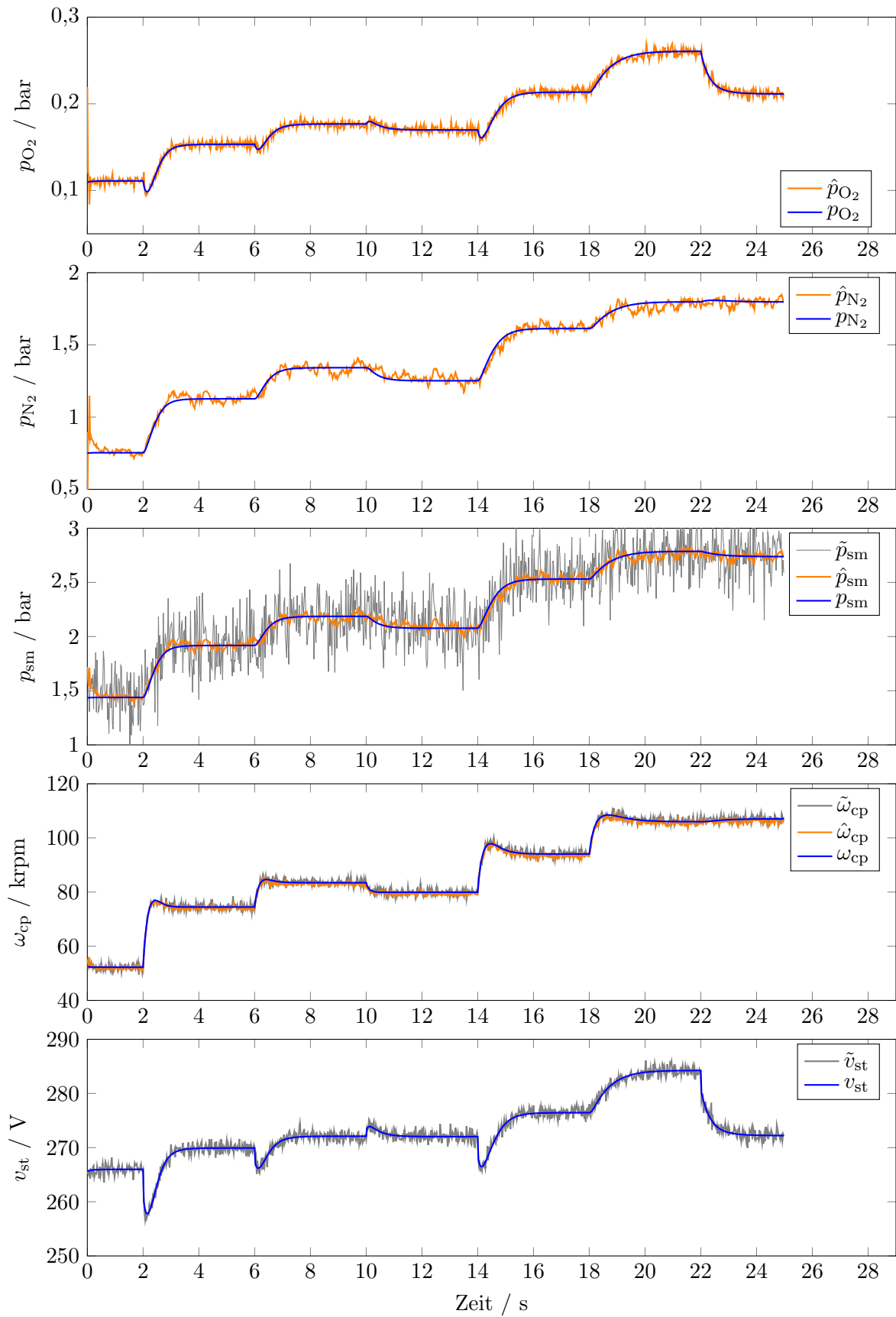


Abbildung 2.19: Simulation des Kalman-Filters (Mit Tilde gekennzeichnete Größen sind gemessene Größen, mit Dach gekennzeichnete Größen sind Schätzungen des Kalman-Filters)

UAP5.2.TUD.3 (Entwurf nominelle MPC) Aus der Beschreibung der Regelziele (UAP5.2.TUD.1) ergibt sich als Ansatz für die Formulierung der MPC folgendes Optimierungsproblem:

$$\min_{u_0, \dots, u_{N-1}} \sum_{k=0}^{N-1} w_I \cdot (I_{\text{st},k} - I_{\text{load}})^2 + \sum_{k=1}^N (\lambda_{\text{O}_2,k} - \lambda_{\text{O}_2,\text{ref}})^2$$

$$x_{k+1} = f_d(x_k, u_k) \quad k = 0, \dots, N-1$$

$$\begin{aligned} 1,5 &\leq \lambda_{\text{O}_2,k} & k &= 1, \dots, N \\ 20 \text{ krpm} &\leq \omega_{\text{cp},k} \leq 105 \text{ krpm} & k &= 1, \dots, N \\ 50 \text{ V} &\leq v_{\text{cm},k} \leq 250 \text{ V} & k &= 0, \dots, N-1 \end{aligned}$$

wobei $f_d(x, u)$ eine passend diskretisierte Form des oben beschriebenen Modells darstellt.

Für die Umsetzung sind allerdings noch ein paar Anpassungen dieses Optimierungsproblems notwendig. Dabei ist auch die Besonderheit zu beachten, dass bei dem vorliegenden Modell ein Durchgriff der Stellgröße I_{st} auf den Sauerstoffüberschuss λ_{O_2} besteht, und letzterer in der Gütefunktion und vor allem auch den Bedingungen des Optimierungsproblem erscheint. Um sicherzustellen, dass die Bedingung immer eingehalten wird, muss hier zwischen dem Wert $\lambda_{\text{O}_2,k}^-$ des Sauerstoffüberschusses unterschieden werden, der sich zum Zeitschritt k mit der Stellgröße $I_{\text{st},k-1}$ ergibt, die im Intervall $[t_{k-1}, t_k)$ auf das System gegeben war, und dem Wert $\lambda_{\text{O}_2,k}^+$, der sich mit der neuen Stellgröße $I_{\text{st},k}$ ergibt, die im Intervall $[t_k, t_{k+1})$ wirkt:

$$\begin{aligned} \lambda_{\text{O}_2,k}^- &: \text{Berechnet mit } I_{\text{st},k-1} \\ \lambda_{\text{O}_2,k}^+ &: \text{Berechnet mit } I_{\text{st},k} \end{aligned}$$

Es muss für beide Werte verlangt werden, dass diese innerhalb des zulässigen Bereiches liegen.

Zusätzlich sind noch folgende Anpassungen des Optimierungsproblems vorgenommen werden, die sich im Laufe des Entwurfs als sinnvoll erwiesen haben:

- Es wird der Gütefunktion ein weiterer Term hinzugefügt, der die Änderung der Verdichterspannung bestraft. Dies ist notwendig, um schwach gedämpfte Schwingungen dieser Stellgröße zu vermeiden.
- Der Sauerstoffüberschuss wird nach oben auf einen Wert von 5 begrenzt. Diese Grenze sollte in der Lösung zwar nie erreicht werden, jedoch hat sich herausgestellt, dass dies dem Optimierer hilft, eine gute Startlösung zu finden.
- Ebenso werden die Drücke nach unten auf 0 Pa begrenzt. Auch diese werden mit einer gültigen Lösung nie erreicht, es hat sich aber gezeigt, dass diese Begrenzungen sich positiv auf die Konvergenz des Optimierers auswirken.

Damit erhält man letztlich das Optimierungsproblem

$$\min_{u_0, \dots, u_{N-1}} \sum_{k=0}^{N-1} w_I \cdot (I_{\text{st},k} - I_{\text{load}})^2 + \sum_{k=1}^N (\lambda_{\text{O}_2,k}^- - \lambda_{\text{O}_2,\text{ref}})^2 + \sum_{k=1}^{N-1} w_{\Delta v} \cdot (v_{\text{cm},k} - v_{\text{cm},k-1})^2$$

$$x_{k+1} = f_d(x_k, u_k) \quad k = 0, \dots, N-1$$

$$\begin{array}{lll}
1,5 \leq \lambda_{\text{O}_2,k}^+ \leq 5 & k = 0, \dots, N-1 \\
1,5 \leq \lambda_{\text{O}_2,k}^- \leq 5 & k = 1, \dots, N \\
20 \text{ krpm} \leq \omega_{\text{cp},k} \leq 105 \text{ krpm} & k = 1, \dots, N \\
50 \text{ V} \leq v_{\text{cm},k} \leq 250 \text{ V} & k = 1, \dots, N \\
0 \leq p_{\text{O}_2,k} & k = 1, \dots, N \\
0 \leq p_{\text{N}_2,k} & k = 1, \dots, N \\
0 \leq p_{\text{sm},k} & k = 1, \dots, N.
\end{array}$$

Für die Gewichte werden die Werte

$$\begin{aligned}
w_{\text{I}} &= 0,05 \\
w_{\Delta v} &= 10^{-5}
\end{aligned}$$

verwendet. Die Abtastzeit beträgt hier $T_{\text{s}} = t_{k+1} - t_k = 0,025 \text{ s}$. Das nichtlineare, zeitkontinuierliche System wird über das explizite Eulerverfahren diskretisiert, welches mit einem Zwischenschritt arbeitet, d. h. mit einer Abtastzeit von $T_{\text{s,RK1}} = 0,0125 \text{ s}$,

$$\begin{aligned}
f_{\text{d}}(x_k, u_k) &= x_{k+1} = x_{k+\frac{1}{2}} + T_{\text{s,RK1}} \cdot f(x_{k+\frac{1}{2}}, u_k) \\
\text{mit } x_{k+\frac{1}{2}} &= x_k + T_{\text{s,RK1}} \cdot f(x_k, u_k)
\end{aligned}$$

Ergebnis der Regelung mit der nominellen MPC Abbildung 2.20 zeigt die Ergebnisse bei einer Folge von Sprüngen des Sollstroms I_{load} . Die ersten beiden Plots zeigen die Stellgrößen, wobei die erste hier auch gleichzeitig die wesentliche Regelgröße darstellt. Man erkennt, dass dem Sollstrom I_{load} fast perfekt gefolgt wird (dies wird gleich noch etwas genauer betrachtet) und die Begrenzung der Verdichterspannung eingehalten wird.

Im dritten Plot ist der Sauerstoffüberschuss und im vierten Plot die Verdichterdrehzahl gezeigt. Auch hier zeigt sich, dass die gegebenen Grenzen immer eingehalten werden. Durch die gewählte Gewichtung in der Gütefunktion ist erreicht, dass die Regelung primär versucht, dem Sollstrom zu folgen, und das Einhalten eines Sauerstoffüberschusses von 2 hier ein nachrangiges Ziel darstellt. Es ist deutlich erkennbar, dass bei jeder Stromänderung der Sauerstoffüberschuss zunächst vom Sollwert von 2 abweicht, bis der Verdichter über die zweite Stellgröße des Systems, der Verdichterspannung v_{cm} , beschleunigt oder abgebremst ist. Im stationären Zustand wird dann wieder ein Sauerstoffüberschuss von 2 erreicht, sofern die maximale Verdichterdrehzahl dies zulässt. Beim letzten Sollstrom in diesem Beispiel ist dies nicht mehr der Fall. Bei der maximalen Verdichterdrehzahl reicht der Luftstrom nicht aus, einen Sauerstoffüberschuss von 2 zu erreichen. Entsprechend stellt sich ein geringerer Wert ein.

Der letzte Plot stellt der Vollständigkeit halber die Drücke als die drei weiteren Zustandsgrößen des Modells dar.

Abbildung 2.21 zeigt den Zeitausschnitt um den ersten Sprung des Sollstroms I_{load} etwas genauer. Hier wird deutlich, dass der Stapelstrom dem Sollstrom nicht direkt folgen kann, da dies zu einem zu großen Einbruch des Sauerstoffüberschusses führen würde. Die MPC begrenzt daher automatisch den Stapelstrom und beschleunigt den Verdichter mit maximaler Spannung, um einen höheren Luftstrom zu erreichen. Der Stapelstrom wird dabei in jedem Schritt so viel wie möglich angehoben, bis dieser nach vier Regelschritten den Sollwert erreicht.

UAP5.2.TUD.4, UAP5.4.TUD.1 (KI-Approximation der MPC und Validierung der eingebetteten MPC) Die beschriebene MPC wurde sowohl mit einem NN als auch einem GP gelernt. Die damit erhaltenen Ergebnisse sind mit einer entsprechenden Analyse im Abschnitt 2.1.1 ab Seite 21 bei der Beschreibung der Ergebnisse zum AP1 beschrieben.

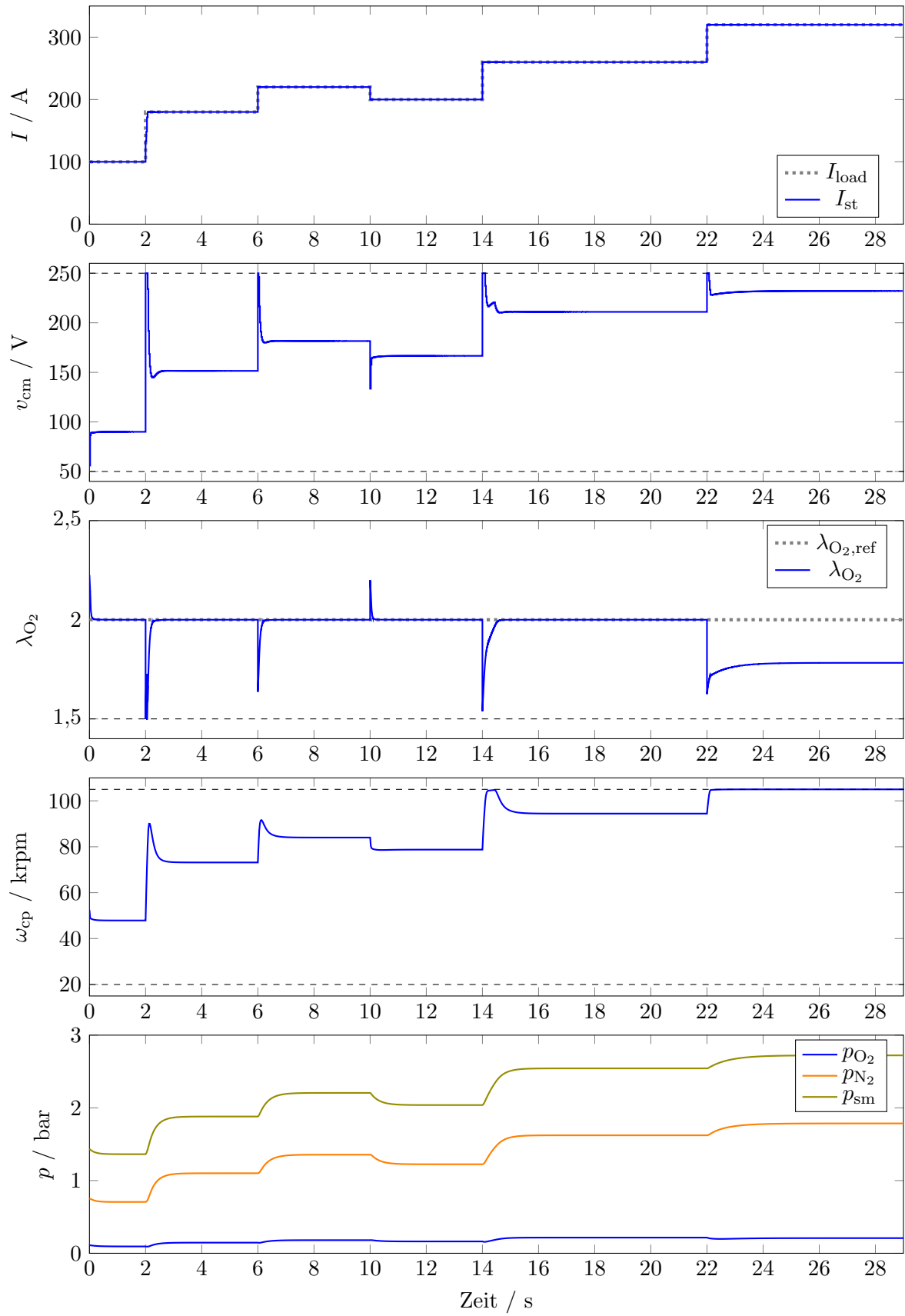


Abbildung 2.20: Ergebnisse der nominellen MPC beim Testfall

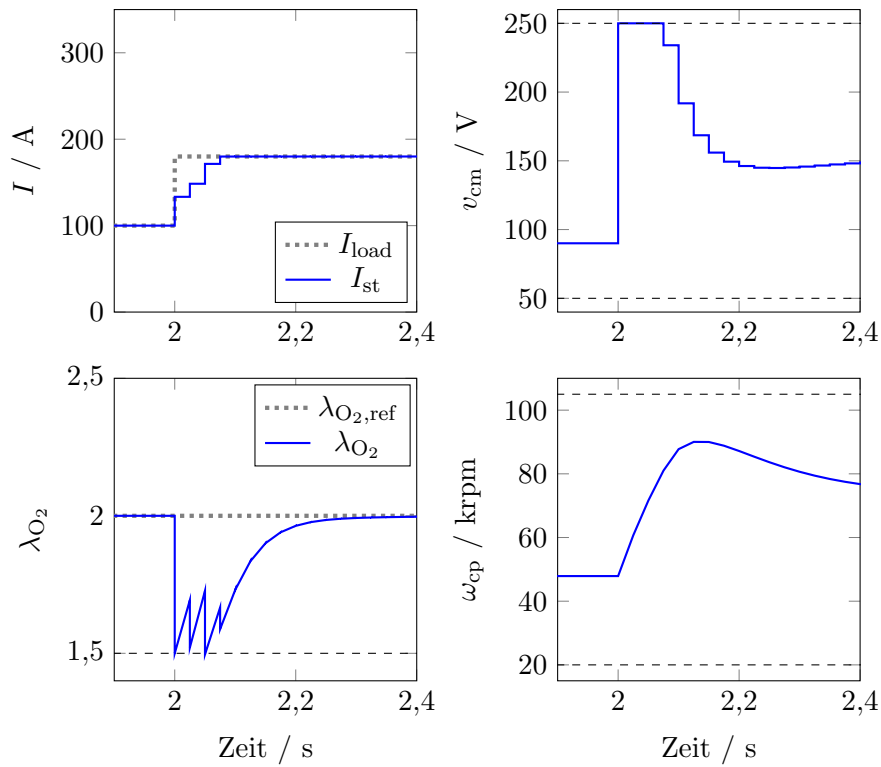


Abbildung 2.21: Ergebnisse der nominellen MPC beim ersten Sprung des Sollstromes im Testfall

UAP5.1.TUD.2 (Open-Source-Bibliothek) Alle Python-Programme, um die hier gezeigten Ergebnisse zu erzeugen, inklusive einer entstandenen Toolbox zu einfachen Modellierung sowie dem Regler- und Beobachterentwurf, ist als Open-Source-Software veröffentlicht und zusammen mit den anderen entstandenen Softwarekomponenten am Ende des Abschnitt 2.1.1 (Seite 27) aufgelistet.

2.2 Wichtigste Positionen des zahlenmäßigen Nachweises

Die vom Projektträger erhaltenen Mittel wurden entsprechend dem Projektantrag eingesetzt, um die unter Abschnitt 2.1 genannten Ergebnisse zu erreichen. Der wesentliche Teil entfiel dabei auf die Personalkosten der wissenschaftlichen Mitarbeiter, die dieses Projekt bearbeitet hatten. Im geringeren Umfang wurden auch wissenschaftliche Hilfskräfte eingesetzt, die Simulationsstudien durchgeführt und an dem entstandenen Programmcode mitgearbeitet haben.

Die erhaltenen Sachmittel entfielen im Wesentlichen auf leistungsfähige Workstations und einen KI-Rechenserver, die für ein effizientes Trainieren der KI-Modelle notwendig waren. Die Reisekosten wurden für die halbjährlichen Treffen mit den Projektpartnern und zum Teil auch Projektträger sowie für die Teilnahme an internationalen Konferenzen zur Vorstellung der erreichten Ergebnisse eingesetzt.

2.3 Notwendigkeit und Angemessenheit der geleisteten Arbeit

Alle durchgeführten Arbeiten folgten dem Projektplan und stellten notwendige und logische Schritte auf dem Weg zum Erreichen des Projektzieles dar.

2.4 Voraussichtlicher Nutzen, insbesondere der Verwertbarkeit des Ergebnisses im Sinne des fortgeschriebenen Verwertungsplans

Die von der TUDa im Rahmen des Projektes durchgeführte Arbeiten haben zu einem Erkenntnisgewinn geführt, der als Basis für die praktische Anwendung von MPC-Ansätzen in Bereichen darstellt, bei denen dies bislang aus Ressourcengründen nicht möglich war. Zudem kann die wissenschaftliche Gemeinschaft auf diesen Ergebnissen weiter aufbauen.

Konkret für die TUDa und das Fachgebiet CCPS ergibt sich durch diese Arbeiten eine Erhöhung der Sichtbarkeit durch die Veröffentlichung der Ergebnisse und der Teilnahme an wissenschaftlichen Konferenzen. Hiervon verspricht sich das Fachgebiet CCPS auch Vorteile bei der Einwerbung weiterer Drittmittelprojekte in dem Bereich der Anwendung von MPC-Ansätzen auf eingebetteten Systemen bzw. allgemeiner auf Systemen mit geringer Rechenleistung.

Durch die Förderung konnten hochqualifizierte Ingenieure ausgebildet werden, die Ihre erworbenen Fähigkeiten in Industrie und Wissenschaft weiter anwenden können. Zudem wurden auch Studierende im Rahmen von studentischen Projekten und Arbeiten einbezogen.

2.5 Während der Durchführung des Vorhabens dem ZE bekannt gewordener Fortschritt auf dem Gebiet des Vorhabens bei anderen Stellen

Im Bereich der Künstlichen Intelligenz wurden während der Laufzeit dieses Projektes enorme Fortschritte gemacht. Dies wird besonders an der rasante Verbreitung einfach verfügbarer KI-Anwendungen, wie z. B. LLMs (Large Language Models) von denen hier nur ChatGPT von OpenAI als ein kommerzielles Beispiel genannt werden soll, deutlich. Jedoch werden damit andere Anwendungen erschlossen, als die, die hier im Rahmen dieses Projektes betrachtet wurden. Der große Ressourcenbedarf der genannten Systeme macht einen Einsatz auf eingebetteten Systemen wie Steuergeräten aktuell noch

nicht vorstellbar. Es stellt für die Zukunft jedoch ein spannendes Forschungsthema dar, ob diese Systeme vorteilhaft im Entwurfsprozess der in diesem Projekt betrachteten Regelungen eingesetzt werden können.

Betrachtet man den enger gefassten Bereich der Kombination von KI mit modellprädiktiver Regelung, so ist auch dort eine gewisse Forschungstätigkeit in den letzten Jahren zu erkennen. Viele Arbeiten verfolgen dabei jedoch andere Schwerpunkte, die sich mit dem in diesem Projekt gewählten Ansatz nur bedingt überschneiden, aber in ihrer Zielsetzung oft komplementär sind. Ein verbreiteter Fokus liegt etwa auf der Anwendung von Deep-Learning-Modellen zur Abbildung hochkomplexer, oft schwer modellierbarer Systemdynamik. Solche Ansätze werden insbesondere im Bereich autonomer Fahrzeuge, bei Robotern oder für chemische Prozesse untersucht, und dabei das Ziel verfolgt, das Verhalten nichtlinearer Systeme durch datengetriebene Modelle möglichst präzise zu erfassen [16]. Typischerweise steht dabei die Modellqualität im Vordergrund, wobei teilweise auch Nachteile hinsichtlich der Echtzeitfähigkeit oder Interpretierbarkeit in Kauf genommen werden [7], [58].

Im Gegensatz dazu lag der Schwerpunkt dieses Projekts auf der effizienten Integration datenbasierter Modelle in MPC-Verfahren unter Berücksichtigung von Echtzeittauglichkeit, Robustheit und struktureller Einfachheit, insbesondere mit Blick auf ressourcenbeschränkte Umgebungen. Gerade hier ergeben sich Anknüpfungspunkte, diese komplementäre Ansätze in Zukunft mit unseren Ergebnissen zu verbinden. So könnten etwa in anderen Projekten entwickelte hochgenaue Blackbox-Modelle, die aufgrund ihrer Komplexität aktuell nicht für den Echtzeiteinsatz geeignet sind, mit unseren echtzeitfähigen Optimierungsverfahren kombiniert werden [36]. Auch Methoden zur automatisierten Generierung von Modellen aus großen Datenmengen könnten genutzt werden, um den Modellierungsprozess in unserem Ansatz weiter zu vereinfachen oder zu skalieren. Damit ließe sich das Ziel verfolgen, die Modellgüte weiter zu erhöhen, ohne die Anforderungen an Berechenbarkeit und Regelungseffizienz zu kompromittieren [7], [18], [36].

Gleichzeitig sind teilweise auch Überschneidungen erkennbar: So beschäftigen sich einige Arbeiten mit der Unsicherheitsquantifizierung in datengetriebenen Modellen für MPC – ein Aspekt, der auch in unserem Projekt aufgegriffen wurde [40]. Ebenso werden zunehmend Methoden untersucht, bei denen KI-Modelle zur Beschleunigung der MPC-Berechnung selbst verwendet werden, etwa durch approximative Optimierung oder durch lernbasierte Näherung an MPC-Lösungen. Diese Ziele decken sich in Teilen mit unseren Bestrebungen, insbesondere was die Reduktion der Rechenzeit auf eingebetteten Plattformen betrifft.

Insgesamt zeigt sich, dass die aktuelle Forschung in diesem Bereich sehr breit gefächert ist. Die im Projekt verfolgten Ansätze ergänzen diese Entwicklungen sinnvoll, indem sie sich auf anwendungsnahe und implementierbare Lösungen konzentrieren, die eine realitätsnahe Nutzung auf technischer Ebene ermöglichen. Die parallelen Entwicklungen unterstreichen dabei die Aktualität des Themas und eröffnen zugleich interessante Perspektiven für zukünftige Kooperationen und methodische Erweiterungen.

2.6 Veröffentlichungen

- [1] H. Alsmeier, A. Savchenko und R. Findeisen, „Neural Horizon Model Predictive Control - Increasing Computational Efficiency with Neural Networks“, in *2024 American Control Conference (ACC)*, 2024, S. 1646–1651. DOI: 10.23919/ACC60939.2024.10644452.
- [2] H. Alsmeier, L. Theiner, A. Savchenko, A. Mesbah und R. Findeisen, „Imitation Learning of MPC with Neural Networks: Error Guarantees and Sparsification“, in *2024 IEEE 63rd Conference on Decision and Control (CDC)*, IEEE, Dez. 2024, S. 4777–4782. DOI: 10.1109/cdc56724.2024.10886138.
- [3] R. Findeisen, A. Rose, K. Graichen und M. Mönnigmann, „Embedded Optimization in Control: An Introduction, Opportunities, and Challenges“, in *Reference Module in Materials Science and Materials Engineering*, Elsevier, 2024, ISBN: 978-0-12-803581-8. DOI: 10.1016/B978-0-443-14081-5.00129-X.
- [4] A. Rose, M. Pfefferkorn, H. H. Nguyen und R. Findeisen, „Learning a Gaussian Process Approximation of a Model Predictive Controller with Guarantees“, in *2023 62nd IEEE Conference on Decision and Control (CDC)*, 2023, S. 4094–4099. DOI: 10.1109/CDC49753.2023.10384047.
- [5] A. Rose, P. Schaub und R. Findeisen, *Safe and High-Performance Learning of Model Predictive Control using Kernel-Based Interpolation*, (angenommen bei American Control Conference 2025), 2024. arXiv: 2410.06771 [eess.SY].
- [6] R. Schurig, A. Himmel und R. Findeisen, „Toward Grassmannian Dimensionality Reduction in MPC“, *IEEE Control Systems Letters*, Jg. 7, S. 3187–3192, 2023. DOI: 10.1109/LCSYS.2023.3291229.

Literatur

- [7] G. I. Beintema, M. Schoukens und R. Tóth, „Deep subspace encoders for nonlinear system identification“, *Automatica*, Jg. 156, S. 111210, 2023. DOI: <https://doi.org/10.1016/j.automatica.2023.111210>.
- [8] A. Bemporad, M. Morari, V. Dua und E. N. Pistikopoulos, „The explicit linear quadratic regulator for constrained systems“, *Automatica*, Jg. 38, Nr. 1, S. 3–20, 2002, ISSN: 0005-1098. DOI: [https://doi.org/10.1016/S0005-1098\(01\)00174-1](https://doi.org/10.1016/S0005-1098(01)00174-1). Adresse: <https://www.sciencedirect.com/science/article/pii/S0005109801001741>.
- [9] J. Bethge, M. Pfefferkorn, A. Rose, J. Peters und R. Findeisen, „Model Predictive Control with Gaussian-Process-Supported Dynamical Constraints for Autonomous Vehicles“, *IFAC-Papers-OnLine*, Jg. 56, Nr. 2, S. 507–512, 2023, 22nd IFAC World Congress, ISSN: 2405-8963. DOI: <https://doi.org/10.1016/j.ifacol.2023.10.1618>. Adresse: <https://www.sciencedirect.com/science/article/pii/S2405896323020268>.
- [10] L. T. Biegler und V. M. Zavala, „Large-scale nonlinear programming using IPOPT: An integrating framework for enterprise-wide dynamic optimization“, *Computers & Chemical Engineering*, Jg. 33, Nr. 3, S. 575–582, 2009.
- [11] V. D. Blondel und J. N. Tsitsiklis, „A survey of computational complexity results in systems and control“, *Automatica*, Jg. 36, Nr. 9, S. 1249–1274, 2000.
- [12] H. G. Bock, *Randwertproblemmethoden zur parameteridentifizierung in systemen nichtlinearer differentialgleichungen*. Rheinische Friedrich-Wilhelms-Universität, 1987.
- [13] H. G. Bock, M. Diehl, D. Leineweber und J. Schlöder, „Efficient direct multiple shooting in nonlinear model predictive control“, *Scientific Computing in Chemical Engineering, SCCE II*, S. 218–227, 1999.
- [14] H. G. Bock und K.-J. Plitt, „A multiple shooting algorithm for direct solution of optimal control problems“, *IFAC Proceedings Volumes*, Jg. 17, Nr. 2, S. 1603–1608, 1984.
- [15] M. C. Campi und S. Garatti, *Introduction to the Scenario Approach*. Society for Industrial and Applied Mathematics, 2018.
- [16] K. Y. Chee, T. Z. Jiahao und M. A. Hsieh, „KNODE-MPC: A Knowledge-Based Data-Driven Predictive Control Framework for Aerial Robots“, *IEEE Robotics and Automation Letters*, Jg. 7, Nr. 2, S. 2819–2826, 2022. DOI: [10.1109/LRA.2022.3144787](https://doi.org/10.1109/LRA.2022.3144787).
- [17] S. Chen, K. Saulnier, N. Atanosov u. a., „Approximating Explicit Model Predictive Control Using Constrained Neural Networks“, in *2018 Annual American Control Conference (ACC) June 27-29, 2018. Wisconsin Center, Milwaukee, USA*, 2018.
- [18] A. B. S. Costa, J. Y. Nishiura, P. V. Lopes, F. A. N. Verri und A. Skoogh, „Artificial Data Generation for Smart Manufacturing Systems: Discrete Event Simulation, Traceability, and Process Mining“, in *Encontro Nacional de Inteligência Artificial e Computacional (ENIAC)*, SBC, 2024, S. 719–730.
- [19] M. A. Danzer, S. J. Wittmann und E. P. Hofer, „Prevention of fuel cell starvation by model predictive control of pressure, excess ratio, and current“, *Journal of Power Sources*, Jg. 190, Nr. 1, S. 86–91, 2009, Selected Papers presented at the 11th ULM ElectroChemical Days, ISSN: 0378-7753. DOI: <https://doi.org/10.1016/j.jpowsour.2008.12.089>. Adresse: <https://www.sciencedirect.com/science/article/pii/S0378775308024749>.

- [20] M. Diehl, H. J. Ferreau und N. Haverbeke, „Efficient numerical methods for nonlinear MPC and moving horizon estimation“, in *Nonlinear model predictive control*, Springer, 2009, S. 391–417.
- [21] M. Diehl, L. Magni und G. De Nicolao, „Efficient NMPC of unstable periodic systems using approximate infinite horizon closed loop costing“, *Annual Reviews in Control*, Jg. 28, Nr. 1, S. 37–45, 2004.
- [22] A. Domahidi, A. U. Zgraggen, M. N. Zeilinger, M. Morari und C. N. Jones, „Efficient interior point methods for multistage problems arising in receding horizon control“, in *2012 IEEE 51st IEEE conference on decision and control (CDC)*, IEEE, 2012, S. 668–674.
- [23] D. T. Doncevic, A. M. Schweidtmann, Y. Vaupel, P. Schäfer, A. Caspari und A. Mitsos, „Deterministic Global Nonlinear Model Predictive Control with Neural Networks Embedded“, *IFAC-PapersOnLine*, Jg. 53, Nr. 2, S. 5273–5278, 2020. DOI: 10.1016/j.ifacol.2020.12.1207.
- [24] Z. Fan, X. Yu, M. Yan und C. Hong, „Oxygen Excess Ratio Control of PEM Fuel Cell Based on Self-adaptive Fuzzy PID“, *IFAC-PapersOnLine*, Jg. 51, Nr. 31, S. 15–20, 2018. DOI: 10.1016/j.ifacol.2018.10.004.
- [25] J. Golbert und D. R. Lewin, „Model-based control of fuel cells: (1) Regulatory control“, *Journal of Power Sources*, Jg. 135, Nr. 1-2, S. 135–151, Sep. 2004. DOI: 10.1016/j.jpowsour.2004.04.008.
- [26] J. Golbert und D. R. Lewin, „Model-based control of fuel cells (2): Optimal efficiency“, *Journal of Power Sources*, Jg. 173, Nr. 1, S. 298–309, Nov. 2007. DOI: 10.1016/j.jpowsour.2007.04.062.
- [27] A. Griewank und A. Walther, *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.
- [28] S. Gros, M. Zanon, R. Quirynen, A. Bemporad und M. Diehl, „From linear to nonlinear MPC: bridging the gap via the real-time iteration“, *International Journal of Control*, Jg. 93, Nr. 1, S. 62–80, 2020. DOI: 10.1080/00207179.2016.1222553. eprint: <https://doi.org/10.1080/00207179.2016.1222553>. Adresse: <https://doi.org/10.1080/00207179.2016.1222553>.
- [29] J. Hensman, N. Fusi und N. D. Lawrence, „Gaussian Processes for Big Data“, *CoRR*, 2013. Adresse: <http://arxiv.org/abs/1309.6835>.
- [30] J. Hensman, A. Matthews und Z. Ghahramani, „Scalable Variational Gaussian Process Classification“, in *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, 9.–12. Mai 2015.
- [31] M. Hertneck, J. Köhler, S. Trimpe und F. Allgöwer, „Learning an Approximate Model Predictive Controller With Guarantees“, *IEEE Control Systems Letters*, Jg. 2, Nr. 3, S. 543–548, 2018. DOI: 10.1109/LCSYS.2018.2843682.
- [32] B. Karg und S. Lucia, „Efficient Representation and Approximation of Model Predictive Control Laws via Deep Learning“, *IEEE Transactions on Cybernetics*, Jg. 50, Nr. 9, S. 3866–3878, Sep. 2020. DOI: 10.1109/TCYB.2020.2999556.
- [33] M. Ławryńczuk, „Neural Networks in Model Predictive Control“, in *Intelligent Systems for Knowledge Management*, Springer Berlin Heidelberg, 2009, S. 31–63. DOI: 10.1007/978-3-642-04170-9_2.
- [34] W. C. Li und L. T. Biegler, „Process control strategies for constrained nonlinear systems“, *Industrial & engineering chemistry research*, Jg. 27, Nr. 8, S. 1421–1433, 1988.
- [35] M. Mangold, A. Bück und R. Hanke-Rauschenbach, „Passivity based control of a distributed PEM fuel cell model“, *Journal of Process Control*, Jg. 20, Nr. 3, S. 292–313, März 2010. DOI: 10.1016/j.jprocont.2009.11.008.
- [36] J. Mun, H.-G. Jo und C. S. Park, „Toward scalable prediction of indoor thermal dynamics: Neural-network-implanted state-space (NNiSS) model“, *Energy and Buildings*, Jg. 331, S. 115 359, 2025. DOI: <https://doi.org/10.1016/j.enbuild.2025.115359>.

- [37] W. K. Na und B. Gou, „Feedback-Linearization-Based Nonlinear Control for PEM Fuel Cells“, *IEEE Transactions on Energy Conversion*, Jg. 23, Nr. 1, S. 179–190, März 2008. DOI: 10.1109/tec.2007.914160.
- [38] J. Nocedal und S. J. Wright, *Numerical optimization*. Springer, 1999.
- [39] T. Ohtsuka, „A continuation/GMRES method for fast computation of nonlinear receding horizon control“, *Automatica*, Jg. 40, Nr. 4, S. 563–574, 2004.
- [40] P. Pauli, A. Koch, J. Berberich, P. Köhler und F. Allgöwer, „Training robust neural networks using Lipschitz bounds“, *IEEE Control Syst. Lett.*, Jg. 6, S. 121–126, 2021.
- [41] J. Pohlodek, H. Alsmeier, B. Morabito, C. Schlauch, A. Savchenko und R. Findeisen, „Stochastic Model Predictive Control Utilizing Bayesian Neural Networks“, in *2023 American Control Conference (ACC)*, ISSN: 2378-5861, Mai 2023, S. 603–608. DOI: 10.23919/ACC55779.2023.10156115. Adresse: <https://ieeexplore.ieee.org/document/10156115>.
- [42] J. Pohlodek, H. Alsmeier, B. Morabito, C. Schlauch, A. Savchenko und R. Findeisen, „Stochastic Model Predictive Control Utilizing Bayesian Neural Networks“, in *2023 American Control Conference (ACC)*, ISSN: 2378-5861, Mai 2023, S. 603–608. DOI: 10.23919/ACC55779.2023.10156115. Adresse: <https://ieeexplore.ieee.org/document/10156115>.
- [43] J. T. Pukrushpan, H. Peng und A. G. Stefanopoulou, „Control-Oriented Modeling and Analysis for Automotive Fuel Cell Systems“, *Journal of Dynamic Systems, Measurement, and Control*, Jg. 126, Nr. 1, S. 14–25, März 2004. DOI: 10.1115/1.1648308.
- [44] J. T. Pukrushpan, „Modelling and Control of Fuel Cell Systems and Fuel Processors“, Diss., University of Michigan, 2003.
- [45] C. E. Rasmussen und C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, Nov. 2005, ISBN: 9780262256834. DOI: 10.7551/mitpress/3206.001.0001. Adresse: <https://doi.org/10.7551/mitpress/3206.001.0001>.
- [46] J. Rawlings, D. Mayne und M. Diehl, *Model predictive control: theory, computation, and design*. Nob Hill Publishing, 2017.
- [47] T. Salzmann, E. Kaufmann, M. Pavone, D. Scaramuzza und M. Ryll, *Neural-MPC: Deep Learning Model Predictive Control for Quadrotors and Agile Robotic Platforms*, 2022. DOI: 10.48550/ARXIV.2203.07747.
- [48] D. Simon, *Optimal State Estimation Kalman, H_∞ , and Nonlinear Approaches*. Hoboken, New Jersey: Wiley-Interscience, 2006.
- [49] E. Snelson und Z. Ghahramani, „Sparse Gaussian Processes using Pseudo-inputs“, in *Advances in Neural Information Processing Systems*, Y. Weiss, B. Schölkopf und J. Platt, Hrsg., Bd. 18, MIT Press, 2005.
- [50] M. C. Steinbach, „A structured interior point SQP method for nonlinear optimal control problems“, in *Computational Optimal Control*, Springer, 1994, S. 213–222.
- [51] K. W. Suh, „Modeling, analysis and control of fuel cell hybrid power systems“, Diss., University of Michigan, 2006.
- [52] M. Titsias, „Variational Learning of Inducing Variables in Sparse Gaussian Processes“, in *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, (16.–18. Apr. 2009), D. van Dyk und M. Welling, Hrsg., Ser. Proceedings of Machine Learning Research, Bd. 5, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA: PMLR, 2009, S. 567–574. Adresse: <https://proceedings.mlr.press/v5/titsias09a.html>.
- [53] A. Vahidi, A. Stefanopoulou und H. Peng, „Model predictive control for starvation prevention in a hybrid fuel cell system“, in *Proceedings of the 2004 American Control Conference*, IEEE, 2004. DOI: 10.23919/acc.2004.1383709.

- [54] R. Verschueren, G. Frison, D. Kouzoupis u. a., „acados – a modular open-source framework for fast embedded optimal control“, *Mathematical Programming Computation*, 2021.
- [55] A. Wächter und L. T. Biegler, „On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming“, *Mathematical Programming*, 2006.
- [56] A. G. Wilson und H. Nickisch, „Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP)“, *CoRR*, Jg. abs/1503.01057, 2015. arXiv: 1503.01057. Adresse: <http://arxiv.org/abs/1503.01057>.
- [57] C. Ziogou, S. Papadopoulou, M. C. Georgiadis und S. Voutetakis, „On-line nonlinear model predictive control of a PEM fuel cell system“, *Journal of Process Control*, Jg. 23, Nr. 4, S. 483–492, Apr. 2013. DOI: 10.1016/j.jprocont.2013.01.011.
- [58] N. Zolman, U. Fasel, J. N. Kutz und S. L. Brunton, *SINDy-RL: Interpretable and Efficient Model-Based Reinforcement Learning*, 2024. eprint: 2403.09110. Adresse: <https://arxiv.org/abs/2403.09110>.